Information technology Applications

ITA

1/2026

PAN-EUROPEAN UNIVERSITY
Faculty of Informatics

VZDELÁVANIE VEDA VÝSKUM
EDUCATION SCIENCE RESEARCH

SSKI
SLOVENSKÁ SPOLOČNOSŤ PRE KYBERNETIKU A INFORMATIKU PRI SAV
SLOVAK SOCIETY FOR CYBERNETICS AND INFORMATICS

ITA - International Journal of Information Technology Applications

Volume 15, Number 1, July 2026

AIMS AND SCOPE OF ITA

The primary aim of the International Journal of Information Technology Applications (ITA) is to publish high-quality papers of new development and trends, novel techniques, approaches and innovative methodologies of information technology applications in the broad areas. The International Journal of ITA is published twice a year. Each paper is refereed by two international reviewers. Accepted papers will be available online with no publication fee for authors. The International Journal of ITA is being prepared for the bibliographic scientific databases.

Editor-in-Chief

prof. Ing. Štefan Kozák, PhD. Faculty of Informatics, Pan-European University in Bratislava
stefan.kozak@paneurouni.com

Executive Editor

Ing. Juraj Štefanovič, PhD., Faculty of Informatics, Pan-European University in Bratislava
juraj.stefanovic@paneurouni.com

Editorial Board

Mikhail A. Basarab, Russia
Ivan Brezina, Slovakia
Yakhua G. Buchaev, Russia
Ivana Budinská, Slovakia
Ján Cigánek, Slovakia
Ladislav Cvetko, Croatia
Gabriela Czanner,
Silvester Czanner,
United Kingdom
Zbigniew Domański, Poland
Maria Jose Escalona, Spain
Andrej Ferko, Slovakia
Vladimír S. Galayev, Russia
Pedro Gamito, Portugal
Ladislav Hudec, Slovakia
Oto Haffner, Slovakia
Matthias Harders, Austria
Adam Herout, Czech Republic
Ladislav Hluchý, Slovakia
Oleg Choporov, Russia
Aleš Janota, Slovakia
Mike Joy, United Kingdom
Gabriel Juhás, Slovakia
Martin Juhás, Slovakia
Jozef Kelemen, Czech Republic
Sergey Kirsanov, Russia
Vladimir I. Kolesnikov, Russia
Štefan Kozák, Slovakia
Alena Kozáková, Slovakia
Vladimír Krajčík, Czech Republic
Erik Kučera, Slovakia
Jaroslav Kultán, Slovakia
Ján Lacko, Slovakia
Igor Lvovich, Russia
Eva Mihaliková, Slovakia
Branislav Mišota, Slovakia
Pavle Mogin, New Zealand
Mykola S. Nikitchenko, Ukraine
Ján Paralič, Slovakia
Rimantas Petrauskas, Lithuania
Lucia Pomello, Italy
Martin Potančok, Czech Republic
Roman Povalej, Germany
Wolf Rauch, Austria
Danica Rosinová, Slovakia
Eugen Ružický, Slovakia
Václav Řepa, Czech Republic
Peter Sinčák, Slovakia
Elena Somova, Bulgaria
Vjeran Strahonja, Croatia
Jiří Šafařík, Czech Republic
Petr Šaloun, Czech Republic
József K. Tar, Hungary
Valentino Vranić, Slovakia
Adam Wojciechowski, Poland
Doina Zmaranda, Romania

ITA - International Journal of Information Technology Applications

Instructions for authors

The International Journal of Information Technology Applications is welcoming contributions related with the journal´s scope. Scientific articles in the range approximately 10 standard pages are reviewed by two international reviewers. Reports up to 5 standard pages and information notices in range approximately 1 standard page are accepted after the decision of editorial board. Contributions should be submitted via e-mail to the editorial office. The language of contributions is English. Text design should preserve the layout of the template file, which may be downloaded from the webpage of journal. Papers in this journal are provided under diamond acess policy (no fee, open website) and authors declare their own Creative Commons license.

Deadlines of two standard issues per year (special issues are possible)

paper submission deadline – continuous process
review decision – continuous process
release date – July/December

Editorial office address

Faculty of Informatics, Pan-European University, Tematínska 10, 851 05 Bratislava, Slovakia
juraj.stefanovic@paneurouni.com

Published by

Pan-European University, Slovakia, https://www.paneurouni.com/
Paneurópska vysoká škola, n.o., Tomášikova 20, 821 02 Bratislava, IČO 36 077 429
Civil Association EDUCATION-SCIENCE-RESEARCH, Slovakia, https://v-v-v.eu/
OZ VZDELÁVANIE -VEDA-VÝSKUM, Andrusovova 5, 851 01 Bratislava,
IČO 42 255 180
Slovak Society for Cybernetics and Informatics (SSKI)
at the Slovak Academy of Sciences, Slovakia
Slovenská spoločnosť pre kybernetiku a informatiku pri SAV (SSKI),
Ústav automobilovej mechatroniky, Fakulta elektrotechniky a informatiky STU,
Ilkovičova 3, 812 19 Bratislava 1, IČO 00 178 730, https://www.sski.sk/

Electronic online version of journal

https://www.itajournal.com/ visit Archive and Instructions for authors:
(Open Journal System)

Electronic backup and preservation

www.webdepozit.sk

digitálne pramene

Print version

Multigrafika s.r.o., Rajecká 13, 821 07 Bratislava

Print issues

Contact the editorial office,
print issues are available until they are in stock.

ISSN: 2453-7497 (online)
ISSN: 1338-6468 (print version) Registration No.: EV 4528/12

Contents

Set of contributions in the convention of students´ scientific/technical works
at the Faculty of Informatics, Pan-European University in Bratislava,
May 14th, 2026

Editorial

Dear readers,

each season in spring, our Faculty of Informatics arranges a contest of students´ scientific/technical works. In previous issues of our journal some of these students´works were included. This time, we proudly dedicate the whole issue 1/2026 for presentations of all participants, they took part in the convention on May 14th, 2026 at our Faculty. Alltogether seven students expressed their interest to participate, including one foreign student who completes here the Erasmus study program for visiting students.

To the next issues, we will thank you for contributions - we welcome your latest research results and technical solutions and we look forward to further collaboration.

Ing. Juraj Štefanovič, PhD.
ITA Executive editor

Final moment of the convention on May 14th, 2026 at our Faculty of Informatics.

International Journal
Information Technology Applications (ITA)

Volume 15, Number 1, July 2026

REAL-TIME DEBUG VISUALIZATION VIA INTER-PROCESS COMMUNICATION IN UNITY EDITOR

Dušan Paulovič

Abstract:

Debugging iterative geometric algorithms in a game engine editor presents a fundamental challenge: when execution is suspended at a breakpoint, the editor becomes non-interactive, making visual inspection of intermediate computational states impossible. This paper presents a debug visualization system that decouples rendering from the main application process through Inter-Process Communication (IPC) via memory-mapped files. Drawing commands - lines, points, meshes, text, and geometric primitives - are serialized using a binary command pattern and transmitted to a secondary Unity Editor instance, which renders them independently while the primary process is suspended. The system exposes a renderer abstraction through the IDrawContext interface, enabling custom rendering backends with no changes to the core system. Zero runtime overhead in production builds is guaranteed by C# conditional compilation. The system was developed and validated in practice during the implementation of geometric collision detection algorithms - including GJK, EPA, SAT, MPR, and incremental convex hull - within a bachelor's thesis project.

Keywords:

Debug visualization, Unity Editor, inter-process communication, memory-mapped file, command pattern, geometric algorithms.

Introduction

Visual debugging is an essential practice in the development of algorithms operating in three-dimensional space. Geometric algorithms such as the Gilbert-Johnson-Keerthi (GJK) distance algorithm, the Expanding Polytope Algorithm (EPA), the Separating Axis Theorem (SAT), and the Minkowski Portal Refinement (MPR) method are inherently spatial: their correctness depends on the relative positions and orientations of geometric structures that are difficult to reason about from numeric output alone. Authoritative treatments of GJK and EPA [1, 2] consistently accompany their algorithmic descriptions with geometric figures, because the correctness of each iteration step can only be assessed visually.

The Unity Editor provides built-in debug drawing utilities, most notably the Gizmos and Handles APIs [3, 4]. These are sufficient for inspecting algorithm state between frames in normal play mode. However, when execution is suspended at a breakpoint, the primary editor instance becomes non-interactive for the purposes of scene inspection [5, 6]. As a result, the developer cannot reliably rotate the scene view, observe newly drawn primitives, or interact with the editor while stepping through an algorithm. This limitation makes visual debugging ineffective at exactly the moments it is most needed.

This paper presents a system that resolves this problem. The core idea is to move rendering to a separate, independent process: a secondary Unity Editor instance dedicated exclusively to visualization. The primary application serializes drawing commands to a named memory-mapped

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

file; the secondary instance reads and renders these commands continuously, remaining fully interactive regardless of the primary process state. The contribution lies not in a new visualization paradigm, but in a practical integration of out-of-process rendering into the Unity Editor debugging workflow.

The system was designed as a practical development tool and validated through active use during the implementation of several geometric collision detection algorithms for a bachelor's thesis on rigid body physics.

1 Related Work

Several tools and approaches exist for visual debugging within the Unity Editor environment, each with limitations relevant to the problem addressed here.

Unity's built-in Gizmos API allows drawing geometric primitives in the Scene View during edit and play modes [3]. The Handles API, part of the UnityEditor namespace, provides enhanced drawing with more stylistic control [4]. Both APIs are editor-only: they cannot be referenced by runtime code compiled into players. More importantly, both operate within the same process as the editor and are therefore subject to the same thread-freezing limitation during breakpoint debugging: they cannot update while execution is paused.

Logging-based approaches record numeric state to the console or to a file. A developer could in principle build a custom visualizer that reads such logs and renders the geometry, which is conceptually similar to the approach presented in this paper. However, this system does so automatically, using a binary format optimized for spatial data, and renders directly into a secondary Unity Editor instance that is already available to every Unity developer on all supported platforms, requiring no additional tooling.

To the author's knowledge, no widely adopted tool within the Unity ecosystem provides live debug drawing via out-of-process rendering that remains interactive while the primary process is suspended at a breakpoint.

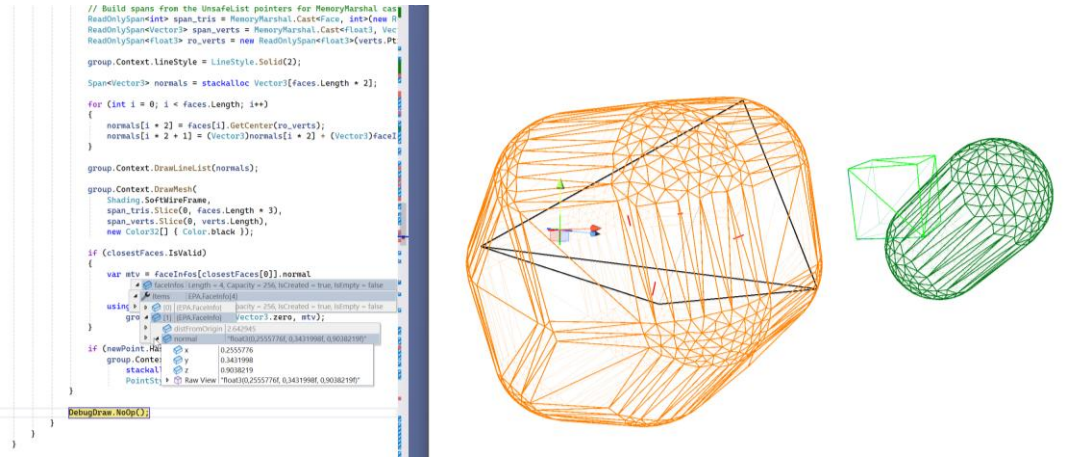

Fig.1. The left side of the image shows Visual Studio paused at a breakpoint, while the right side displays an interactive visualization of the Minkowski difference between a box and a capsule, with the initial simplex of the EPA algorithm contained within it.

2 System Architecture

The system consists of two components: the Producer and the Consumer, communicating through a shared memory region as illustrated in (Fig.2).

The Producer component runs in the developer's primary Unity Editor instance. It consists of the static DebugDraw API class, which accepts drawing calls from application code; the CommandSerializer, which translates drawing calls into binary command data; and the FileCommandHandler with its MMFHandler, which writes serialized commands to a named memory-mapped file.

The Consumer component runs in a secondary Unity Editor instance hosting a dedicated visualization project, identified by the DBG_ prefix in its project name. It consists of the FileCommandDrawer, which periodically reads from the shared memory; the CommandAggregator, which organizes commands by index and group; and the UnityEditorDrawingContext, which executes rendering via Unity's GL API in the Scene View.

Data flows in one direction: primary application → CommandSerializer → memory-mapped file → FileCommandDrawer → CommandAggregator → UnityEditorDrawingContext → Scene View. The secondary instance remains fully interactive, allowing scene rotation and inspection, regardless of the primary process state.

The secondary instance is launched via the Visual Debug / Open Instance editor menu item, which copies the necessary assets into a temporary directory and spawns a new Unity process pointing to that directory. The secondary project automatically detects the DBG_ prefix at startup and initializes the rendering server component (DebugDrawFileServer). Once started, the consumer runs as a long-lived visualization process that persists throughout the debugging session, independent of the execution state of the primary editor. It is started once and reused across recompilations, domain reloads, and debugger attach and detach cycles, amortizing initialization cost across the entire session.

The Unity Editor is reused as a visualization runtime rather than reimplemented, trading higher resource usage for zero additional tooling requirements and immediate integration into the existing development environment. Every Unity developer already has the Unity Editor installed, and it provides scene navigation, camera controls, GL rendering, and cross-platform support without any additional dependencies.

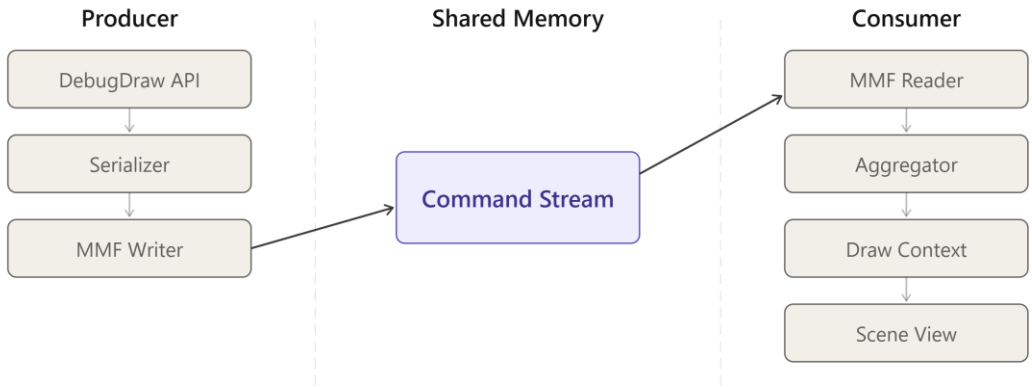

Fig.2. System architecture: Producer serializes commands to shared memory. Consumer reads and renders them independently.

3 Technical Design

3.1 Command Abstraction and Binary Serialization

All drawing operations are represented as commands. The ICommand interface extends ICustomBytes, a serialization contract requiring three members: the ByteSize property, and Serialize and Deserialize methods operating on byte spans. This design cleanly separates command identity from its wire format.

Serialization is performed through a fluent chain of MemoryIterator method calls. MemoryIterator maintains an integer offset into a Span<byte> buffer and provides typed Write and Read methods covering all primitive types as well as specialized handlers for Vector3, Quaternion, Matrix4x4, and UTF-8 strings. This approach is allocation-free, avoids reflection, and produces compact binary representations with predictable sizes.

For example, DrawLineCmd is a 32-byte unmanaged struct containing a CmdHeader (8 bytes: command index and an Operation enum value) and two Vector3 values (12 bytes each). Its serialization chain is (Fig.3):

```
public int ByteSize =>
    MemoryUtil.MoveBySizeOf<CmdHeader>().
    MoveBySizeOfVector3(2);

public int Deserialize(ReadOnlySpan<byte> source) =>
    MemoryUtil.
    Read(source, ref h).
    ReadVector3(source, ref s).
    ReadVector3(source, ref e);

public int Serialize(Span<byte> dest) =>
    MemoryUtil.
    Write(dest, h).
    WriteVector3(dest, s).
    WriteVector3(dest, e);
```

Fig.3. DrawLineCmd serialization.
MemoryUtil operates on MemoryIterator,
which is implicitly convertible to int.

Commands with variable-length payload, DrawLineStripCmd (variable point count) and DrawTextCmd (UTF-8 string), compute ByteSize dynamically using the same MemoryIterator chain in a sizing pass before serialization. This maintains a single source of truth for the byte layout.

Binary serialization was preferred over text-based formats [7] for three reasons: lower serialization latency, smaller memory footprint per command, and deterministic byte layout enabling safe use of unmanaged struct operations and stackalloc buffers. As the producer and consumer are always executed from the same build within a single development environment, version compatibility between command layouts is not enforced: structural changes propagate simultaneously to both sides.

3.2 Renderer Abstraction via IDrawContext

The IDrawContext interface defines the complete drawing vocabulary of the system: DrawLine, DrawLineStrip, DrawLineList, DrawCircleFraction, DrawSphere, DrawWireSphere, DrawCube, DrawWireCube, DrawPointList, DrawText, DrawText3D, and DrawMesh. It also exposes state properties (color, matrix, lineStyle) and scope-based state management methods (SetColorScope, SetMatrixScope, SetLineStyleScope, SetAppearanceScope) that restore the previous state when disposed.

Two IDrawContext implementations are directly involved in the presented Unity Editor pipeline. CommandSerializer translates each method call into a serialized command written to the IPC channel. UnityEditorDrawingContext executes GL calls directly in the Unity Scene View of the consumer process.

```
public interface IDrawContext
{
    IDisposable scope { get; set; }
    Color32 color { get; set; }
    Matrix4x4 matrix { get; set; }
    LineStyle lineStyle { get; set; }

    IDisposable SetFrameScope();
    IDisposable SetMatrixScope(in Matrix4x4 matrix);
    IDisposable SetColorScope(Color32 color);
    IDisposable SetLineStyleScope(LineStyle lineStyle);
    IDisposable SetAppearanceScope(Color32 color, LineStyle lineStyle);

    void SetAppearance(Color32 color, LineStyle lineStyle);
    void DrawLine(in Vector3 p1, in Vector3 p2);
    void DrawCircleFraction(in Vector3 center, float radius,
       in Quaternion rotation, float fraction);
    void DrawLineStrip(ReadOnlySpan<Vector3> points, bool looped);
    void DrawLineList(ReadOnlySpan<Vector3> points);
    void DrawSphere(in Vector3 center, float radius);
    void DrawWireSphere(in Vector3 center, float radius);
    void DrawCube(in Vector3 center, in Vector3 halfExtents, in Quaternion rotation);
    void DrawWireCube(in Vector3 center, in Vector3 halfExtents, in Quaternion rotation);
    void DrawPointList(ReadOnlySpan<Vector3> points, PointStyle pointStyle);
    void DrawText(in Vector3 position, string text, TextStyle style);
    void DrawText3D(in Vector3 position, in Quaternion rotation, string text, TextStyle style);
    void DrawMesh(Shading shading, ReadOnlySpan<int> triangles,
        ReadOnlySpan<Vector3> vertices, ReadOnlySpan<Color32> colors);
}

public interface IAppContext
{
    void Clear();
    void BeginGroup(string name);
    void ClearGroup(string name);
}

public interface IBatchContext : IDrawContext, IAppContext
{
    IDisposable BatchScope();
}
```

Fig.4. Interfaces IDrawContext, IAppContext and IBatchContext.

This separation means that the same user code works identically whether it calls into CommandSerializer (in the primary process) or directly into UnityEditorDrawingContext (in a hypothetical single-process configuration). It also allows developers to implement custom rendering backends by providing an IDrawContext implementation and connecting it to a FileCommandDrawer or CommandAggregator. Examples of possible custom backends include: an SVG exporter for documentation, a recording context for playback debugging (already implemented for unit testing purposes), or a runtime renderer for non-editor builds.

The IDrawContext interface is further extended by IAppContext, which provides Clear, BeginGroup, and ClearGroup operations. IBatchContext combines both, enabling the CommandSerializer to serve as the single entry point for all operations from user code (Fig.4).

3.3 Inter-Process Communication via Memory-Mapped Files

The IPC channel is implemented as a named memory-mapped file (MMF) managed by MMFHandler. The MMF has a fixed capacity of 4 MB, configurable via a single constant. Its layout consists of a 4-byte integer length prefix followed by a UTF-8 encoded sequence of newline-separated hex strings, each representing one serialized command. Although commands are serialized into compact binary form internally, they can be represented as human-readable strings in multiple formats for debugging purposes. Pure hexadecimal is used during debug sessions for full inspectability, while a mixed format (hexadecimal header followed by Base64-encoded payload) is used in production to balance readability with compactness (Fig.5). This representation was chosen deliberately to make the debug stream human-inspectable, easily persistable, and simple to recover from disk. Serialization and transport are not the primary performance bottlenecks in the intended debugging workflow. The current transport format is therefore sufficient for interactive workloads, where the system remains usable even under heavy command loads, and is not intended for high-frequency or high-volume data streaming.

Hexadecimal (debugging purposes):
```
7B00000009000000CDCC8C3FCDCC0CC033335340CDCC8CC00000B0403333D3C0
```

Base64 (not used, but available as an option):
```
ewAAAAkAAADNzIw/zcwMwDMzU0DNzIzAAACwQDMz08A=
```

Header hexadecimal, rest Base64 (production ready):
```
7B00000009000000zcyMP83MDMAzM1NAzcyMwAAAsEAzM9PA
```

Fig.5. Same DrawLineCmd serialized in alternative formats for readability comparison.

Named memory-mapped files were selected over named pipes and TCP sockets for several reasons [8, 9]. MMFs support multiple concurrent readers without protocol overhead or connection management. Their content persists across writer restarts, meaning the visualization remains visible if the primary application reconnects after a crash or reload. Random-access reading is efficient, requiring no sequential scanning.

Both read and write access are synchronized using a NamedMutex. The mutex is acquired for the duration of each read and write operation, ensuring that the consumer always receives a consistent snapshot of the command stream. Because the consumer reads in the editor update loop and write operations are brief, contention is negligible in practice.

On disposal, MMFHandler persists the last written content to disk as a plain text file. This allows the visualization state to survive both primary and secondary editor restarts.

3.4 Command Grouping and Incremental Updates

The CommandAggregator on the consumer side organizes received commands into named groups via BeginGroup and ClearGroup operations, and maintains a SortedDictionary<int, DrawCmdInfo> indexed by command index. This allows selective invalidation of visualization state: when an algorithm's state changes, only the commands belonging to a specific group need to be cleared and replaced.

The ClearGroup operation removes only the commands associated with a given group name from the sorted dictionary, leaving all other visualization intact. This is useful in situations where a temporary visual state needs to be drawn on top of an existing scene and remain visible only within a specific section of code. A recurring pattern across multiple algorithms is creating a local group around each iteration step, so that only the current intermediate state is replaced while persistent annotations remain visible. For example, in GJK the current simplex replaces the previous one each iteration, in EPA the evolving polytope is updated step by step, in MPR the current portal is refreshed, and in SAT the currently tested separating axis is highlighted and cleared before the next one is tested. In (Fig.1) orange Minkowski difference hull and origin cross are persistent while black simplex and red normals are temporary.

3.5 Zero-Overhead Production Builds

All public methods in the DebugDraw static class are decorated with the [Conditional("DEBUG_DRAW")] attribute as shown in (Fig.6). When the DEBUG_DRAW compilation symbol is absent, as in release builds, the C# compiler eliminates all call sites at compile time, producing no runtime overhead: no method calls, no allocations, and no conditional branches. The debug visualization system is completely invisible to the runtime in production.

Additionally, helper methods in algorithm implementations may check Debugger.IsAttached before executing visualization code, ensuring that even in debug builds the visualization is skipped unless a debugger is actively connected, avoiding any performance impact during normal profiling sessions.

```
[Conditional("DEBUG_DRAW")]
public static void DrawRay(Vector3 start, Vector3 direction)
{
    if (GetContext(out var ctx))
        ctx.DrawLine(start, start + direction * _rayLength);
}

[Conditional("DEBUG_DRAW")]
public static void DrawCircleFraction(
    Vector3 center,
    float radius,
    Quaternion rotation,
    float fraction)
{
    if (GetContext(out var serializer))
        serializer.DrawCircleFraction(center, radius, rotation, fraction);
}
```

Fig.6. DebugDraw methods decorated with Conditional attribute.

4 Rendering Pipeline in UnityEditorDrawingContext

This section describes the rendering pipeline provided by the editor-specific implementation of the IDrawContext interface. All geometry is submitted through Unity's GL immediate-mode interface as triangle lists. There are no persistent Mesh objects, no MeshRenderer components, and no render texture intermediaries, every frame data is constructed on the CPU and pushed directly to the GPU. The entry point is a SceneView callback that fires once per Scene View repaint. Within that callback, a single Material.SetPass(0) call activates the custom shader, and all subsequent GL.Begin / GL.Vertex / GL.End calls are batched into a single draw submission per geometric mode change.

The choice of immediate mode is deliberate [10]. Because the geometry may change every frame and is driven by external data, maintaining persistent Mesh objects would introduce unnecessary complexity: meshes would need to be reallocated or resized on every update, vertex buffers would need to be tracked per object, and the Unity object lifecycle would add overhead. Immediate mode bypasses all of this at the cost of re-submitting every vertex each frame, which is acceptable given that debug overlays are not performance-critical paths.

4.1 The Shader

The entire renderer relies on a single shader with one pass. The vertex shader performs only the standard clip-space transform via UnityObjectToClipPos, and the fragment shader returns the per-vertex color unchanged, making it a pure conduit for whatever geometry and color data the CPU produces. The relevant render-state declarations are:

```
Blend SrcAlpha OneMinusSrcAlpha
ZWrite [_ZWrite]
ZTest LEqual
ZClip False
Cull Off
```

Alpha blending is essential because thick lines are drawn as flat quads that can overlap each other and scene geometry; without it, overlapping segments would produce hard edges at their boundaries. ZClip False prevents Unity from discarding geometry at the near plane: a line segment whose endpoint lies behind the camera would otherwise be partially clipped, but for debug visualization such segments should remain fully visible as outgoing rays. Cull Off is required because a line quad that is nearly edge-on to the camera has a surface normal that can flip relative to the view direction; backface culling would then discard the triangle entirely. ZWrite is exposed as a shader property so that it can be toggled per draw call.

4.2 Camera Abstraction and Unit Conversions

Rendering operates on a minimal camera abstraction that is populated from the active scene Unity camera each frame. It provides a stable interface for accessing view and projection parameters during rendering.

4.2.1 The CameraInfo Snapshot

Rather than querying the camera object repeatedly during a frame (which involves managed property accesses and potential Unity-side validation overhead) all camera state is captured once into an unmanaged CameraInfo struct at the beginning of each repaint. The struct stores the camera's

localToWorldMatrix, the six frustum planes, and the scalar properties orthographic, orthographicSize, fieldOfView, and pixelHeight. Direction vectors and position are read directly from the columns of the stored matrix using unsafe pointer arithmetic, returning read-only references to the actual memory occupied by each column and avoiding any Vector3 copy construction.

4.2.2 World-Space Size of a Screen Unit

The renderer supports four measurement units: World (raw world-space units), Pixel (screen pixels), DIP (device-independent pixels, 1 DIP = DPI/96 pixels), and Em (typographic units, 1 Em = DPI/72 pixels). To draw a line of thickness 2 DIP or text of size 14 Em at any depth in the scene, the system must know the world-space extent corresponding to one such unit at the specific point being drawn.

For a perspective camera, the derivation follows from the standard pinhole projection model [10]. The visible height of the frustum slice at depth z (measured along the camera forward axis) is $H(z) = 2z \cdot \tan(FoV/2)$. Since this height spans pixelHeight pixels, the world-space size of one pixel is:

$$unitsPerPixel(z) = 2z \cdot \tan(FoV / 2) / pixelHeight$$

In code, z is the projection of the vector from the camera position to the query point onto the camera forward axis. For an orthographic camera the formula simplifies to $unitsPerPixel = 2 \cdot orthographicSize / pixelHeight$, independent of depth. The final value is multiplied by the pixel-equivalent of the requested unit type (DPI/96 for DIP, DPI/72 for Em). A line specified as “2 DIP thick” therefore subtends exactly 2 device-independent pixels on screen regardless of its world-space depth, making overlays visually stable under any camera movement or zoom.

4.3 Thick Line Rendering

A one-pixel line can be drawn with GL.LINES, but the effective line width is not a reliable cross-platform mechanism for producing arbitrary thick lines [11]. Producing a line of arbitrary visual thickness therefore requires explicit geometry. Each line segment is replaced by a camera-facing quad: a flat rectangle whose long axis aligns with the segment direction and whose short axis is perpendicular to both the segment direction and the current view direction. The two segment endpoints define the centerline; at each endpoint a perpendicular offset vector is computed, and the four quad corners are the endpoints each displaced by ±offset:

```
quad[0] = A + offset_A      quad[1] = B + offset_B
quad[3] = A − offset_A      quad[2] = B − offset_B
```

This quad is submitted as two triangles: (0, 1, 2) and (0, 2, 3). The offset magnitude equals halfThickness at each endpoint, which is derived from the line thickness value converted to world space via the unit conversion described in Section 4.2.2.

4.3.1 Computing the Perpendicular Offset

The offset vector must be perpendicular to the segment as it appears on screen and must lie in a plane visible to the camera, so that it actually produces width when rasterized. For a perspective camera the view direction at a world point is the normalized vector from the camera position to that point — not the global camera forward, which would introduce directional error for points away from screen centre. Given the local view direction and the segment direction, the screen-space perpendicular is computed by projecting the segment direction onto the plane perpendicular to the view direction and taking the cross product with the view direction:

```
Vector3 projected = Vector3.ProjectOnPlane(dir, viewForward);
Vector3 offset    = Vector3.Cross(projected, viewForward)
                      .normalized * halfThickness;
```

ProjectOnPlane(v, n) computes $v - (v \cdot \hat{n})\hat{n}$, removing the component of the direction vector along the view axis. The cross product of the projected direction with the view direction is then perpendicular to both — pointing sideways on screen regardless of the 3D orientation of the line.

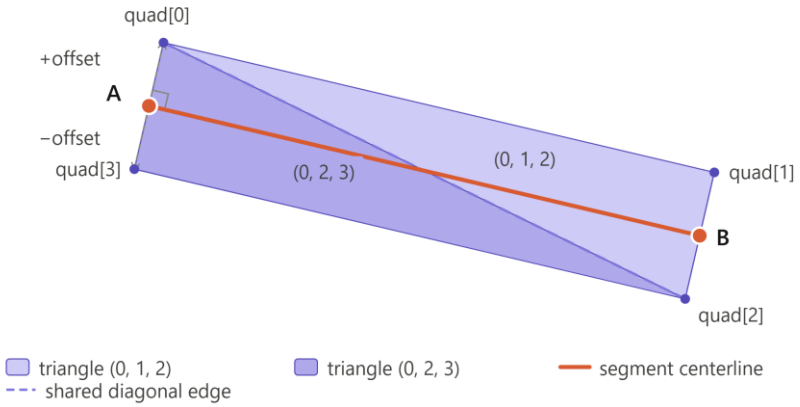

Fig.7. Two triangles per quad, one quad per segment.

4.3.2 Miter Joins

When two segments share a vertex in a polyline, a naive approach produces a gap or overlap between adjacent quads depending on the join angle. A miter join [12] resolves this by extending both quad edges until they intersect at a single point, filling the corner exactly. At a join vertex with incoming projected direction d_A and outgoing direction d_B, the miter direction is the normalised bisector:

```
Vector3 tangent = (dirA + dirB).normalized;
Vector3 miter   = Vector3.Cross(tangent, viewForward).normalized;
```

The miter offset length equals halfThickness / cos θ, where θ is the half-angle between the miter and either segment normal. This follows from the requirement that the miter endpoint lies on both quad edges simultaneously as shown in (Fig.8). For shallow angles $\cos\theta \approx 1$ and the miter approaches halfThickness; for sharp corners $\cos\theta \to 0$ and the length grows without bound. To prevent arbitrarily long spikes, the length is clamped to a configurable multiple of halfThickness; if exceeded, the join falls back to a bevel where both quads terminate at their own segment normals, leaving a small triangular gap that is visually acceptable for corners sharper than roughly 10–15°. A further degenerate case arises when the two projected directions are nearly antiparallel (±180° hairpin): the tangent vector has near-zero magnitude and cannot be normalised, so the outgoing segment normal is used directly.

4.3.3 Segments Facing the Camera

When a segment is nearly parallel to the view direction — pointing roughly toward or away from the camera — it projects to near-zero screen length and the cross-product offset computation

becomes numerically unstable. The condition is detected by testing whether the squared cosine of the angle between the segment direction and view direction is close to 1. When this holds, no quad is emitted; instead, both endpoints are rendered as filled circles with radius equal to halfThickness. This is geometrically correct: a cylindrical line of finite radius viewed end-on appears as a disc of that radius.

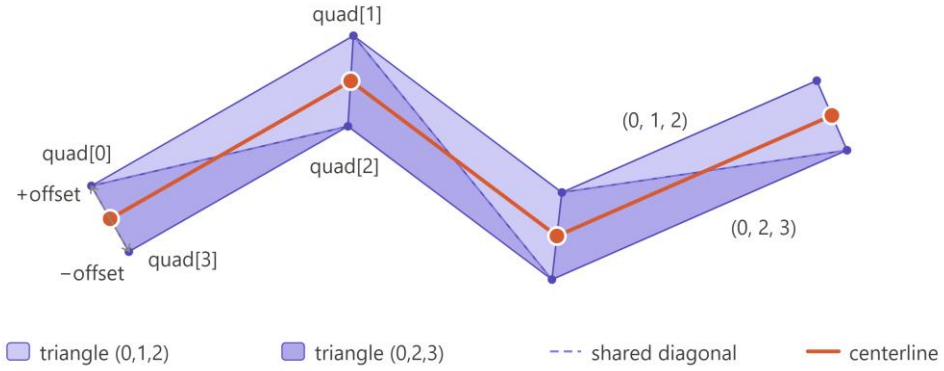

Fig.8. Adjacent quads share miter points at polyline joints.

4.3.4 Round Endpoint Caps

At the two terminal endpoints of a polyline there is no adjacent segment to form a join with. A filled disc is drawn at each endpoint as a triangle fan: its vertices are precomputed as a fixed set of points on the unit circle oriented to face the camera, then scaled by halfThickness and translated to the endpoint position. The same routine is reused for the degenerate end-on segment case (Section 4.3.3) and for rendering Dot pattern elements in dashed lines (Section 4.3.5).

4.3.5 Dashed and Dotted Patterns

Dashed patterns cannot be applied at the segment level of a polyline because the dash/gap rhythm must be consistent in perceived screen distance regardless of how the underlying polyline is sampled [12]. The approach is to first resample the polyline at a uniform interval and then map the resulting point sequence to pattern elements. Resampling operates either in world space (for World-unit spacing) or in screen space (for Pixel and DIP spacing). Screen-space resampling projects each segment to 2D screen coordinates, inserts new sample points at uniform pixel intervals, and recovers the corresponding 3D world position from the interpolated screen position and linearly interpolated depth. This ensures that dashes of a given DIP length maintain constant visual size at any scene depth. The resampled points are consumed by a state machine: a Dot element draws a circle; a Dash element draws a quad spanning two consecutive points; a LongDash element draws two connected quads spanning three points. The pattern repeats cyclically and is encoded in a single byte using two bits per element for up to four elements. For example, the DashDot line pattern is encoded as the value 6 (binary 00 00 01 10), where 10 represents Dash, 01 represents Dot, and 00 terminates the pattern, as shown in (Fig.9).

4.4 Text Rendering

Text is rendered using pre-triangulated vector fonts rather than signed distance field textures or rasterised bitmap atlases. Each glyph is stored as a set of 2D vertices and a triangle index list that

defines the filled area of the glyph outline. Fonts are distributed as GZip-compressed binary files, one per typeface variant, and parsed on first access into an in-memory dictionary keyed by Unicode character. Glyph coordinates are normalised to an emSize of 100 — the cap height of a standard uppercase letter spans 100 units — so the scale factor at render time is the requested font size divided by 100, multiplied by the world-space extent of one screen unit at the text anchor point (Section 4.2.2). Each vertex coordinate axis is stored as an unsigned byte (0–255), scaled by the glyph's own bounding box dimension: coord = raw / 255 × max, where max is the per-glyph width or height stored in the file header, as shown in (Fig.7).

Table 1. Predefined line patterns and their binary encoding.

Name	e[3] e[2] e[1] e[0]	Byte	Preview
Solid	00 00 00 00	0x00	———
Dot	00 00 00 01	0x01	• • • • • • • •
Dash	00 00 00 10	0x02	— — — — — —
DashDot	00 00 01 10	0x06	—• —• —• —•
DashDotDot	00 01 01 10	0x16	—••—••—••—
LongDash	00 00 00 11	0x03	—— —— —— ——
LongDashDot	00 00 01 11	0x07	——•——•——•—
LongDashDotDot	00 01 01 11	0x17	——••——••——••—
LongDashDash	00 00 10 11	0x0B	—— — —— — ——
LongDashDashDotDash	10 01 10 11	0x9B	—— —•— —— —•—

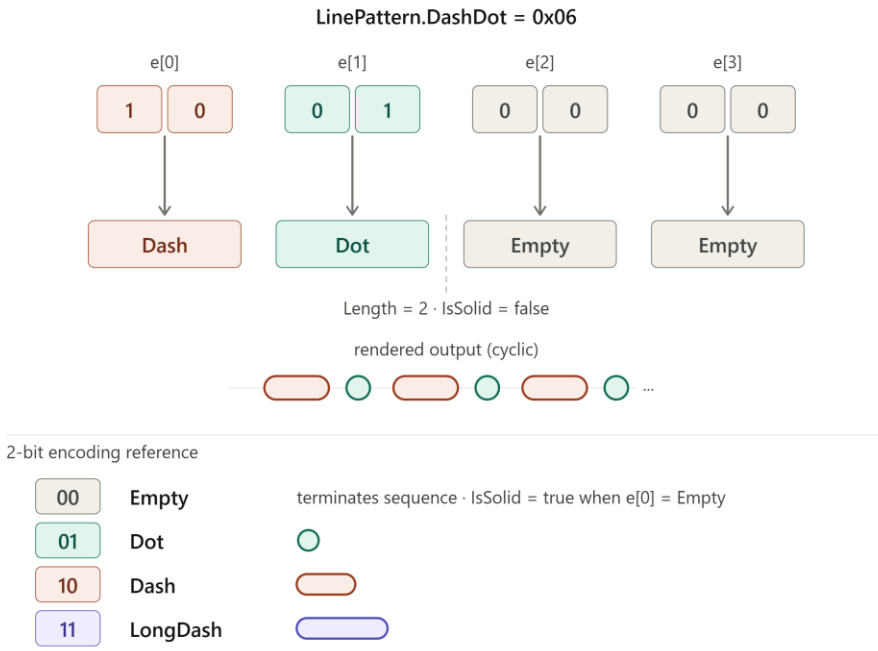

Fig.9. Line patterns are represented by a single byte consisting of up to four two-bit elements.

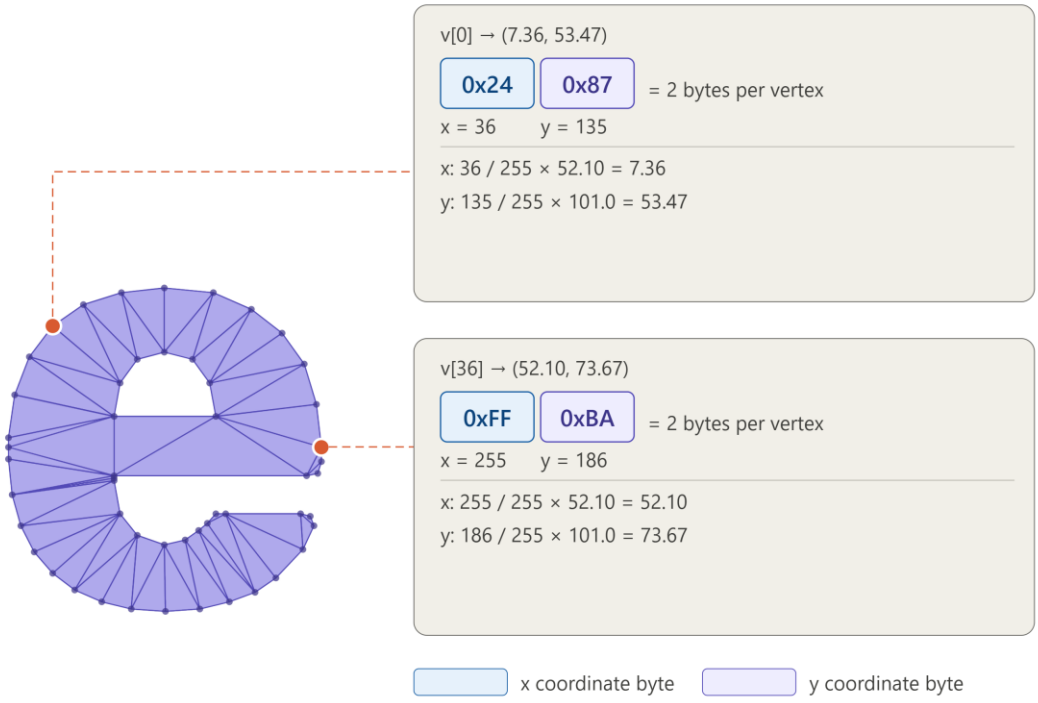

Fig.10. Each vertex coordinate axis is stored as an unsigned byte (0–255).

4.4.1 Layout

Before any vertex data is produced, a measurement pass iterates over the input string and accumulates the total bounding box of the text block. Each glyph contributes its stored width multiplied by the scale factor, plus a configurable letter-spacing value. Space characters advance by the average glyph width; tab characters snap to the next multiple of four average widths; newline characters reset the horizontal position and increment the line counter. From the computed bounds and the style alignment flags, a base offset is derived that positions the text block so the anchor point corresponds to the requested alignment corner or edge. An additional alignment offset scalar pushes the block further along the alignment direction by a fixed screen-space amount, which is useful for preventing a label from overlapping the annotated point.

4.4.2 Glyph Rendering

Each glyph's vertices are transformed from the font's local 2D coordinate system into world space. The y-axis is negated and offset by the glyph height because font coordinate systems place the origin at the top-left with y increasing downward. In billboard mode, the rotation applied to each vertex is taken from the camera transform, ensuring the text plane always faces the viewer. In 3D mode, an explicit rotation quaternion is supplied by the caller. Because the scale depends only on the anchor point, all glyphs within a string are scaled uniformly and text does not deform under perspective. The fill is rendered by iterating over the triangle list and emitting three coloured vertices per triangle directly to the GL stream.

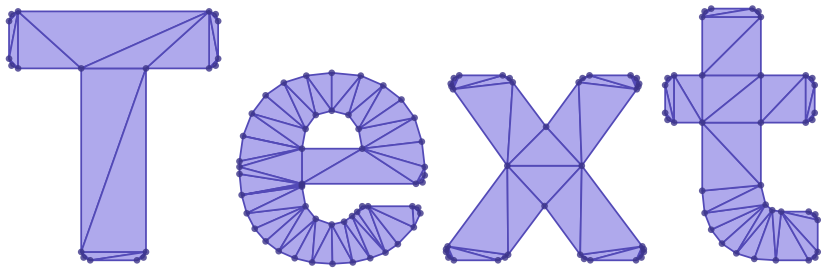

Fig.11. Word "Text" rendered from the binary font format.
Each glyph is decomposed into triangles; vertices are shown as dark dots.

4.4.3 Vector Outline

When an outline width greater than zero is requested, the glyph outline is derived analytically from the triangulation rather than stored separately. The boundary edges of the triangulation — edges belonging to exactly one triangle — collectively form all contour curves of the glyph. They are identified by counting edge occurrences across all triangles: each canonical edge (smaller vertex index first) is counted, and edges with a count of one are boundary edges. These are assembled into closed contours by building an adjacency list and performing an iterative graph walk starting from each unvisited boundary vertex. Each contour — an outer boundary or interior hole — is submitted to GLDrawPolyline as a closed polyline with the outline colour and stroke width, inheriting the full miter join and round cap logic from Section 4.3. Outline quality is resolution-independent since it derives from the same vector data as the fill.

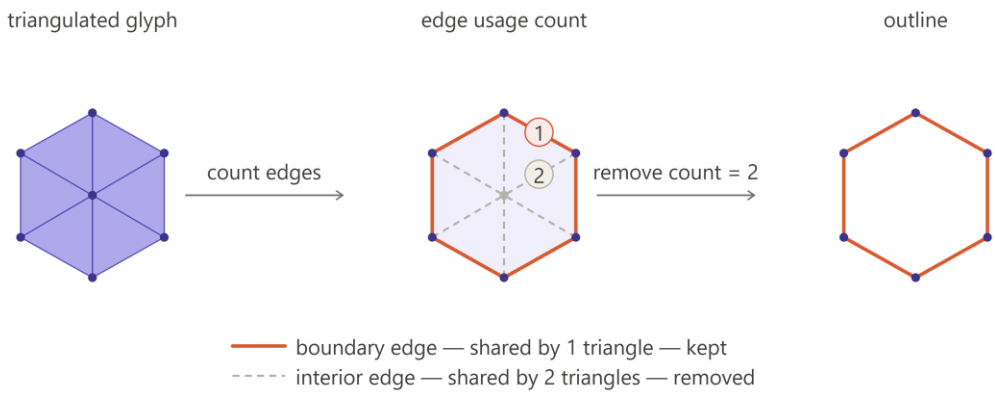

Fig.12. The diagram shows how a glyph outline is extracted from a triangulation.

4.5 Mesh Rendering

Mesh rendering is exposed through a dedicated method for submitting indexed triangle geometry to the renderer:

```
void DrawMesh(
    Shading shading,
    ReadOnlySpan<int> triangles,
    ReadOnlySpan<Vector3> vertices,
    ReadOnlySpan<Color32> colors);
```

This method defines the contract for passing triangle index data together with per-vertex positions and colors. The triangles buffer encodes indexed geometry in triplets, while vertices and colors provide the corresponding attributes. The shading parameter selects the shading mode applied during rasterization. The implementation is responsible for assembling triangles, evaluating their visibility and orientation, and applying the selected shading model in a consistent manner.

4.5.1 Shading Modes

Three shading modes are supported, following the classical taxonomy [10]. Unlit mode renders only front-facing triangles with uniform vertex colours and no lighting computation. A triangle is front-facing when the dot product of its geometric normal with the view direction from the camera to the triangle centroid is positive; the per-triangle view direction is computed individually to account for perspective. Flat shading uses the same dot product as a per-triangle brightness multiplier applied to the vertex colours, producing a faceted appearance useful for inspecting surface orientation. Smooth shading derives per-vertex normals automatically by welding vertices that share the same world-space position within a configurable threshold and accumulating face normals from all adjacent triangles within each welding group. After normalisation, these welded normals produce smooth brightness interpolation across triangulated surfaces without any additional data from the caller.

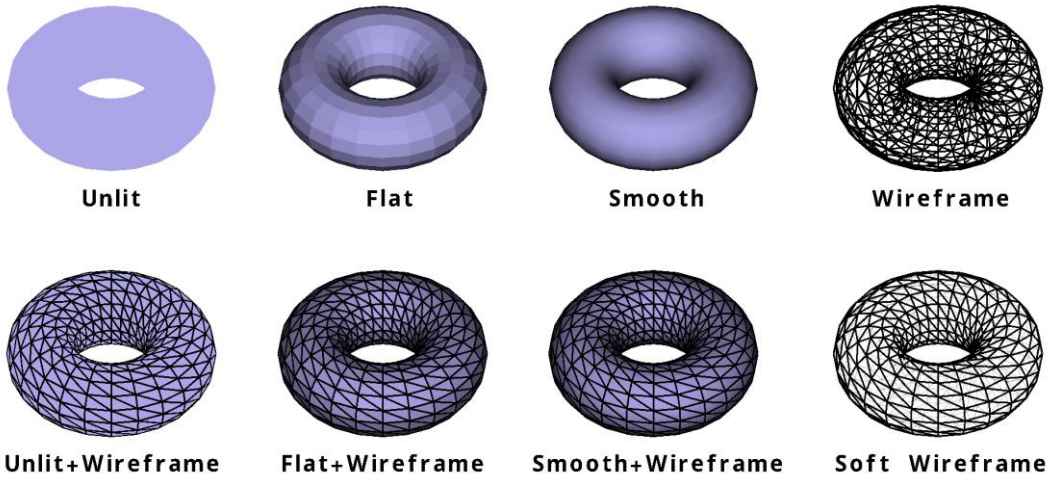

Fig.13. Preview of available mesh smoothing modes.

4.5.2 Wireframe Overlay

When the WireFrame flag is set, mesh edges are rendered as thick lines drawn slightly in front of the surface to avoid z-fighting. The edge extraction step enumerates all triangle edges in canonical form (smaller vertex index first), tags each with a backface flag based on the facing of its parent triangle relative to the camera, and sorts the resulting array. A single linear scan then retains only the first occurrence of each unique edge:

```
Array.Sort(edgesArr.Array, 0, edgeCount);
for (int i = 0; i < edgeCount; i++)
    if (i == 0 || edge.value != edges[i-1].value)
        edges[uniqueCount++] = edge;
```

When the SoftBackfaces flag is enabled, back-facing edges are drawn at 50 % opacity and half the nominal line thickness. This reveals the rear structure of a mesh as a visually subordinate wireframe without obscuring the front, giving an X-ray-like appearance.

5 Limitations and Future Work

The current functionality is limited to simple 3D geometric primitives and text. Rendering features such as 2D graphics in screen coordinates, 3D text with depth, textures, and more complex mesh operations are not yet supported. The IDrawContext abstraction provides a clear extension point for adding these capabilities. The system is designed for developer debugging workflows and is not intended as a general-purpose visualization or rendering framework at this stage.

A planned improvement is the integration of Unity Burst Compiler for performance-critical rendering paths on the consumer side, or alternatively the use of a dedicated external C++ library for these workloads. Burst compilation can significantly reduce the overhead of processing large numbers of draw commands by compiling C# code to highly optimized native code using an LLVM-based optimization pipeline [13]. The implementation is already compatible with the Burst compiler, however, Burst is distributed as a separate package of approximately 500 MB. Since the consumer instance is created as a fresh Unity project, including Burst would substantially increase installation size and setup time. This requires a solution for sharing compiled libraries between the parent project and the consumer instance without redundant installation.

An alternative approach using a native C++ DLL would result in a much smaller footprint. However, the drawback is that the library would need to be compiled separately for each target platform, which increases build and maintenance complexity.

A further planned feature is the ability to share the entire scene state from the primary project to the consumer instance, not just explicit debug draw calls, but the full set of GameObjects, transforms, and meshes present in the primary scene at the time of a breakpoint. This would allow the developer to inspect the complete scene context without any manual instrumentation. This feature requires additional research into what scene data can be efficiently serialized and transmitted via the existing IPC channel, and what Unity APIs are available for reconstructing scene state in the consumer instance.

A standalone native viewer, independent of the Unity Editor, is also in early development. This would allow the consumer to run as a lightweight application rather than a full Unity Editor instance, reducing startup time and resource consumption. The standalone viewer would still be implemented as a Unity player, but its managed code would be compiled ahead-of-time into native code using IL2CPP, eliminating JIT compilation at runtime [14]. As a result, the solution would not require the Unity Editor at runtime. However, the Unity Burst Compiler may still be employed for performance-critical workloads, particularly where data-oriented optimizations such as SIMD vectorization provide measurable benefits.

6 Conclusion

This paper presented a debug visualization system for Unity that decouples rendering from the main application process using inter-process communication via memory-mapped files. The system enables live, interactive visual inspection of algorithmic state during breakpoint debugging,

a capability not provided by existing in-process debug drawing tools, which freeze alongside the application thread. The primary contribution lies in integrating out-of-process rendering into the Unity Editor debugging workflow in a way that requires no additional tooling beyond the Unity Editor itself.

The key technical contributions are: a binary command serialization framework with a clean ICustomBytes contract and allocation-free MemoryIterator chains; a renderer abstraction through the IDrawContext interface enabling custom rendering backends; an IPC channel via named memory-mapped file with mutex-synchronized writes; command grouping with incremental invalidation for efficient state updates; and zero production overhead through [Conditional] attribute-based elimination.

The system was validated in practice as an essential tool during the implementation and adaptation of established geometric collision detection algorithms, including GJK, EPA, SAT, and MPR, as well as an incremental convex hull used as a ground-truth reference, as part of a bachelor's thesis. The collision detection algorithms were implemented independently, with development informed by standard algorithmic descriptions [1,2,10] and, where necessary, by publicly available reference materials and implementations. The incremental convex hull implementation was based on standard algorithmic principles [15] and partially derived from a publicly available reference implementation [16], which influenced its structure and naming. The visual feedback it provided was directly instrumental in identifying and resolving correctness issues that would have been significantly harder to diagnose through numeric logging alone.

Acknowledgement

The author would like to express his sincere gratitude to Ing. Juraj Štefanovič, PhD. for his professional guidance and valuable advice during the development of the bachelor's thesis project in which this system was created. The author also wishes to express his appreciation to his family for their patience, understanding, and support during the completion of this project.

References

[1] Ericson, C. (2005). Real-Time Collision Detection. Morgan Kaufmann. ISBN 978-1-55860-732-3.
[2] van den Bergen, G. (2003). Collision Detection in Interactive 3D Environments. Morgan Kaufmann. ISBN 978-1-55860-801-6.
[3] Unity Technologies (2025). Gizmos and Handles. Unity Manual. Retrieved April 2026, from https://docs.unity3d.com/Manual/gizmos-and-handles.html
[4] Unity Technologies (2025). Handles. Unity Scripting API. Retrieved April 2026, from https://docs.unity3d.com/ScriptReference/Handles.html
[5] Stall, M. (2008). Why you sometimes get a bogus ContextSwitchDeadLock MDA under the debugger. Microsoft Developer Blog. Retrieved April 2026, from https://learn.microsoft.com/en-us/archive/blogs/jmstall/why-you-sometimes-get-a-bogus-contextswitchdeadlock-mda-under-the-debugger
[6] Microsoft Corporation (2024). Debugging Managed Code Using the Windows Debugger. Microsoft Learn. Retrieved April 2026, from https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/debugging-managed-code
[7] Google LLC (2024). Protocol Buffers Developer Guide. Retrieved April 2026, from https://protobuf.dev/overview/
[8] Microsoft Corporation (2024). Memory-Mapped Files. .NET Documentation. Retrieved April 2026, from https://learn.microsoft.com/en-us/dotnet/standard/io/memory-mapped-files
[9] Stevens, W. R., Fenner, B., & Rudoff, A. M. (2004). UNIX Network Programming, Vol. 2: Interprocess Communications (2nd ed.). Addison-Wesley. ISBN 978-0-13-081081-6.

[10] Akenine-Möller, T., Haines, E., & Hoffman, N. (2018). Real-Time Rendering (4th ed.). A K Peters/CRC Press. ISBN 978-1-138-62700-0.
[11] Kilgard, M. J., & Bolz, J. (2012). GPU-accelerated Path Rendering. ACM Transactions on Graphics, 31(6). https://doi.org/10.1145/2366145.2366191
[12] W3C (2011). Scalable Vector Graphics (SVG) 1.1 (Second Edition), Section 11.4. Retrieved April 2026, from https://www.w3.org/TR/SVG11/painting.html
[13] Unity Technologies (2025). Burst Compiler Documentation. Retrieved April 2026, from https://docs.unity3d.com/Packages/com.unity.burst@latest
[14] Unity Technologies (2025). Scripting Backends – IL2CPP. Retrieved April 2026, from https://docs.unity3d.com/Manual/scripting-backends-il2cpp.html
[15] Voxagon (2013). Convex Hulls Revisited. Retrieved April 2026, from https://blog.voxagon.se/2013/03/24/convex-hulls-revisited.html
[16] SpexGuy (n.d.). MinkowskiHull3D. GitHub repository. Retrieved April 2026, from https://github.com/SpexGuy/MinkowskiHull3D

Authors

Dušan Paulovič
Pan-European University,
Faculty of Informatics, Bratislava, Slovakia
dusan.paulovic@gmail.com
Software engineer focused on developer tools and complex desktop applications. He works primarily with C++ and C#, specializing in code parsing, IDE integration, and user interface design. He has contributed to widely used development tooling, with experience spanning CAD/GIS systems, 3D visualization, and performance-oriented software design.

International Journal
Information Technology Applications (ITA)
Volume 15, Number 1, July 2026

MANAGEMENT OF DOCTOR ON-CALL SERVICES

Kristián Štefanka

Abstract:

The article focuses on the design, implementation, and evaluation of a web-based system for doctor shift scheduling. The aim was to create an efficient and user-friendly application that simplifies the scheduling process and reduces administrative workload. The system enables doctors to specify their availability preferences through an interactive calendar, while schedules are generated automatically based on predefined rules. The results show a significant improvement in scheduling efficiency, reduction of errors, and increased user satisfaction compared to traditional Excel-based approaches.

Keywords:

Shift scheduling, healthcare systems, web application, PHP, MySQL, FullCalendar, heuristic algorithm, optimization.

Introduction

The process of scheduling medical staff in healthcare institutions is a complex and demanding task that requires careful coordination of multiple factors [7]. Hospitals operate continuously, and therefore it is necessary to ensure that sufficient medical personnel are available at all times. The quality of scheduling has a direct impact on patient care, employee satisfaction, and the overall efficiency of healthcare services.

In practice, many healthcare institutions still rely on traditional scheduling methods, such as paper-based plans or spreadsheet applications like Microsoft Excel. While these approaches may be sufficient for small teams, they become increasingly inefficient and error-prone as the number of employees and scheduling constraints grows. Manual scheduling requires significant time and effort, and it is difficult to ensure that all constraints are consistently respected.

Modern information technologies provide new opportunities for improving this process. Web-based applications allow centralized data management, real-time updates, and accessibility from multiple devices. These systems can incorporate advanced algorithms that automate scheduling and reduce the likelihood of human error.

The main objective of this paper is to design and implement a web-based system for doctor shift scheduling. The system enables doctors to specify their availability preferences and allows administrators to generate schedules automatically. The solution aims to improve efficiency, reduce administrative workload, and ensure fair distribution of shifts among staff members.

1 Theoretical Background – Doctor Shift Scheduling

The process of scheduling doctor shifts in healthcare institutions represents a complex organizational and optimization problem that directly affects the quality of patient care, staff satisfaction, and the overall efficiency of medical services. Unlike standard workforce planning,

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

hospital scheduling must ensure continuous operation, which requires careful coordination of personnel across different time periods, departments, and medical specializations.

In modern healthcare environments, scheduling is influenced by a wide range of factors. These include the availability of doctors, their qualifications and specializations, legal regulations concerning working hours, and the need to maintain a balanced workload among staff members. In addition, personal preferences of doctors play an important role, as they contribute to employee satisfaction and long-term sustainability of the workforce.

One of the fundamental challenges in shift scheduling is the need to satisfy multiple constraints simultaneously. These constraints may include maximum working hours per week, mandatory rest periods between shifts, and the requirement to have a sufficient number of doctors present at all times. The complexity of these requirements makes manual scheduling increasingly difficult, especially in larger departments.

Traditionally, scheduling has been performed using paper-based systems or spreadsheet applications such as Microsoft Excel. While these approaches are simple and widely accessible, they are often inefficient and prone to human error. Manual updates, lack of automation, and difficulty in enforcing constraints lead to inconsistencies and increased administrative workload.

From a theoretical perspective, doctor shift scheduling can be classified as a combinatorial optimization problem [7]. The goal is to assign a set of doctors to a set of shifts in such a way that all constraints are satisfied while optimizing certain criteria, such as fairness or workload distribution. Exact optimization methods are often computationally expensive, which is why heuristic approaches are commonly used in practice.

Modern scheduling systems aim to address these challenges by utilizing digital tools and algorithms. Web-based applications provide centralized data storage, real-time updates, and improved accessibility. By integrating user preferences and automated decision-making processes, these systems significantly enhance the efficiency and reliability of scheduling.

2 System Design and Scheduling Approach

The design of the proposed system focuses on creating an efficient and user-friendly solution for managing doctor shift schedules. The main objective was to develop a system that allows intuitive interaction for users while ensuring accurate and reliable scheduling results.

The system is based on a web application architecture that enables centralized access for all users. Doctors can log into the system and specify their availability preferences using an interactive calendar. These preferences are categorized into three main states: preferred (“want to work”), unavailable (“cannot work”), and neutral (“can work”). This simple classification allows the system to process user input efficiently while maintaining flexibility.

The scheduling process is designed to minimize manual intervention. Administrators are responsible for overseeing the system and generating schedules, but the majority of the decision-making is handled automatically by the algorithm. This approach reduces the risk of human error and significantly speeds up the scheduling process.

A key aspect of the system design is the implementation of user roles. Two primary roles are defined: doctor and administrator. Doctors have limited access and can only manage their own preferences, while administrators have full control over the system, including user management and schedule generation. This separation ensures security and prevents unauthorized modifications. The overall system architecture is illustrated in (Fig.1).

Another important component is the database structure, which stores all relevant data, including user information, preferences, and generated schedules. The database is designed to maintain consistency and integrity of data, ensuring that all scheduling operations are performed correctly. The database structure of the system is illustrated in (Fig.2).

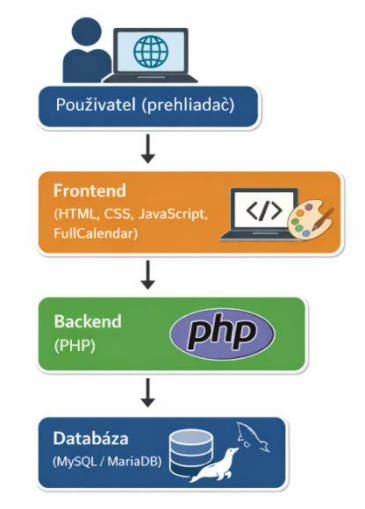

Fig.1. System architecture of the web-based scheduling application.

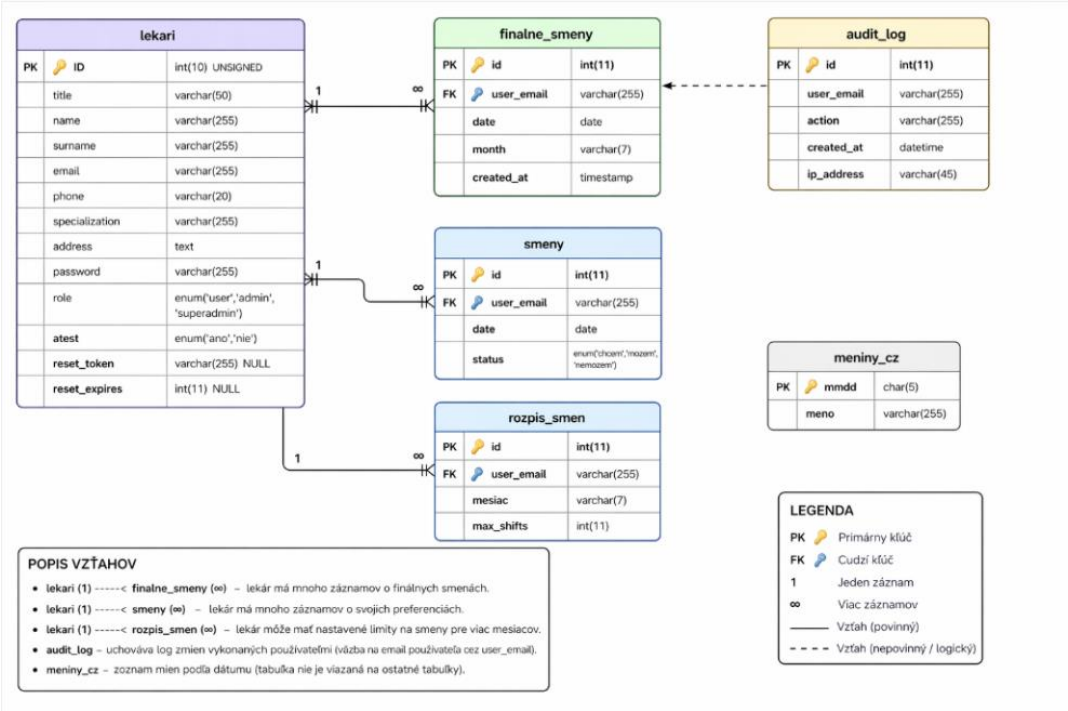

Fig.2. Entity-relationship diagram of the system database.

The scheduling approach is based on a heuristic algorithm [7] that prioritizes simplicity and ef- ficiency. Instead of searching for an optimal solution, the algorithm generates a feasible solution that satisfies all constraints and provides a fair distribution of shifts. This method is particularly suitable for real-world applications, where speed and reliability are more important than theoretical optimality. The scheduling process is illustrated in (Fig.4).

3 Tools Used and Implementation

The implementation of the system is based on widely used web technologies that ensure reliability, scalability, and ease of maintenance. The selection of these technologies was guided by their compatibility, performance, and availability.

The backend of the application is implemented using the PHP programming language [2], which handles server-side logic, processes user requests, and communicates with the database. PHP was chosen due to its simplicity, widespread support, and strong integration with web technologies.

The database layer is managed using MySQL [3], a relational database system that provides efficient data storage and retrieval. The database is designed to support multiple users and ensure data consistency even under concurrent access.

On the client side, the system uses HTML and CSS for structure and design, while JavaScript [4] provides interactivity. The Bootstrap framework [5] is used to create a responsive user interface that adapts to different screen sizes, ensuring usability on both desktop and mobile devices.

A key component of the system is the FullCalendar library [1], which provides an interactive calendar interface. This tool allows users to easily select and modify their availability preferences, offering a visual representation of scheduling data. User interactions and system functionality are illustrated in (Fig.3).

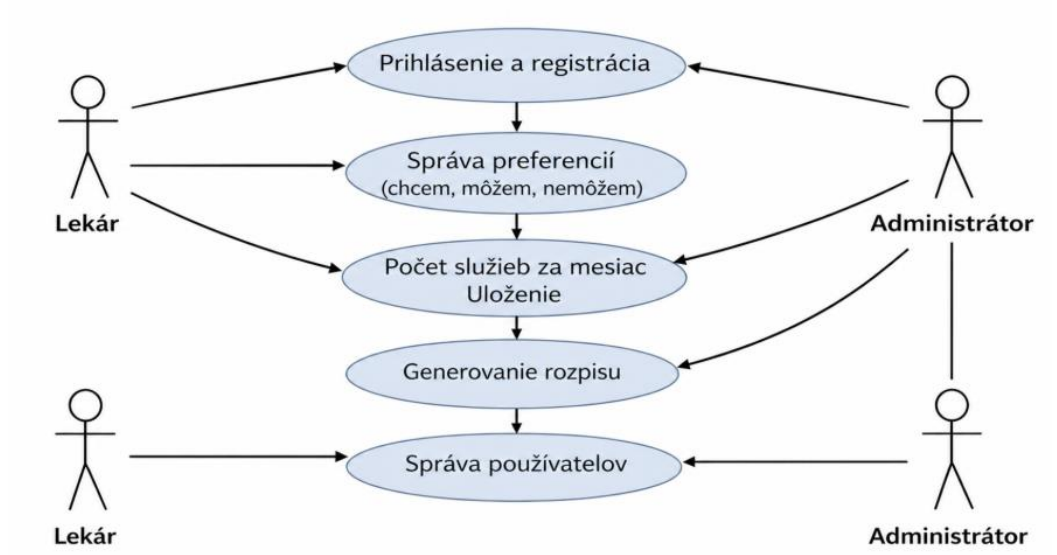

Fig.3. Use case diagram illustrating interactions between doctor and administrator.

Communication between the frontend and backend is implemented using AJAX [6], which enables asynchronous data exchange without reloading the page. This significantly improves user experience and system responsiveness.

The scheduling algorithm is implemented on the server side and processes data stored in the database. It evaluates user preferences and applies predefined rules to generate a schedule. The results are then stored and displayed to users in real time.

In addition to scheduling, the system also includes functionality for exporting data to Excel format. This feature is implemented using a dedicated library and allows administrators to generate reports for further analysis or archival purposes.

An example of the user interface is shown in (Fig.5).

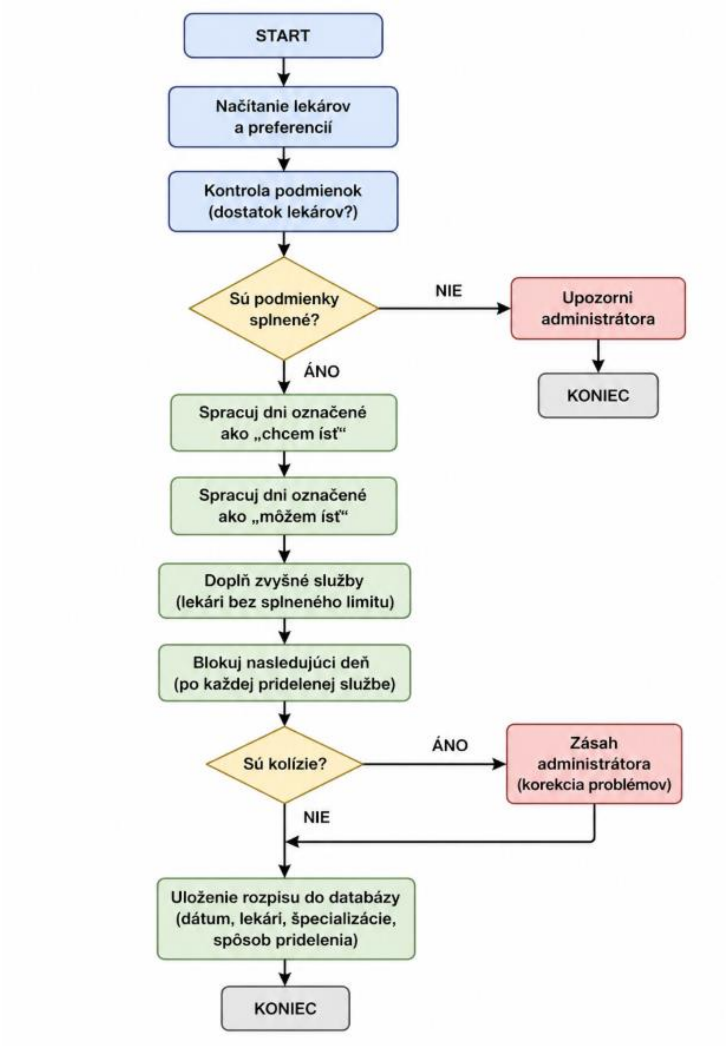

Fig.4. Flowchart of the scheduling algorithm.

4 Analysis of Results

The evaluation of the system represents a crucial part of the development process, as it provides deeper insights into its performance, reliability, and overall effectiveness in real-world conditions. The main objective of the analysis was to determine whether the system meets its intended goals, particularly in terms of efficiency, accuracy, and usability, and to compare its performance with traditional scheduling methods commonly used in healthcare institutions.

The evaluation was conducted using several test scenarios with varying numbers of users and different scheduling conditions. These scenarios were designed to simulate real-world usage, ranging from small departments with a limited number of doctors to larger environments with higher complexity. Key performance indicators included scheduling time, number of errors, system responsiveness, and user satisfaction. These metrics provide a comprehensive overview of the system's capabilities and allow for objective comparison with existing approaches.

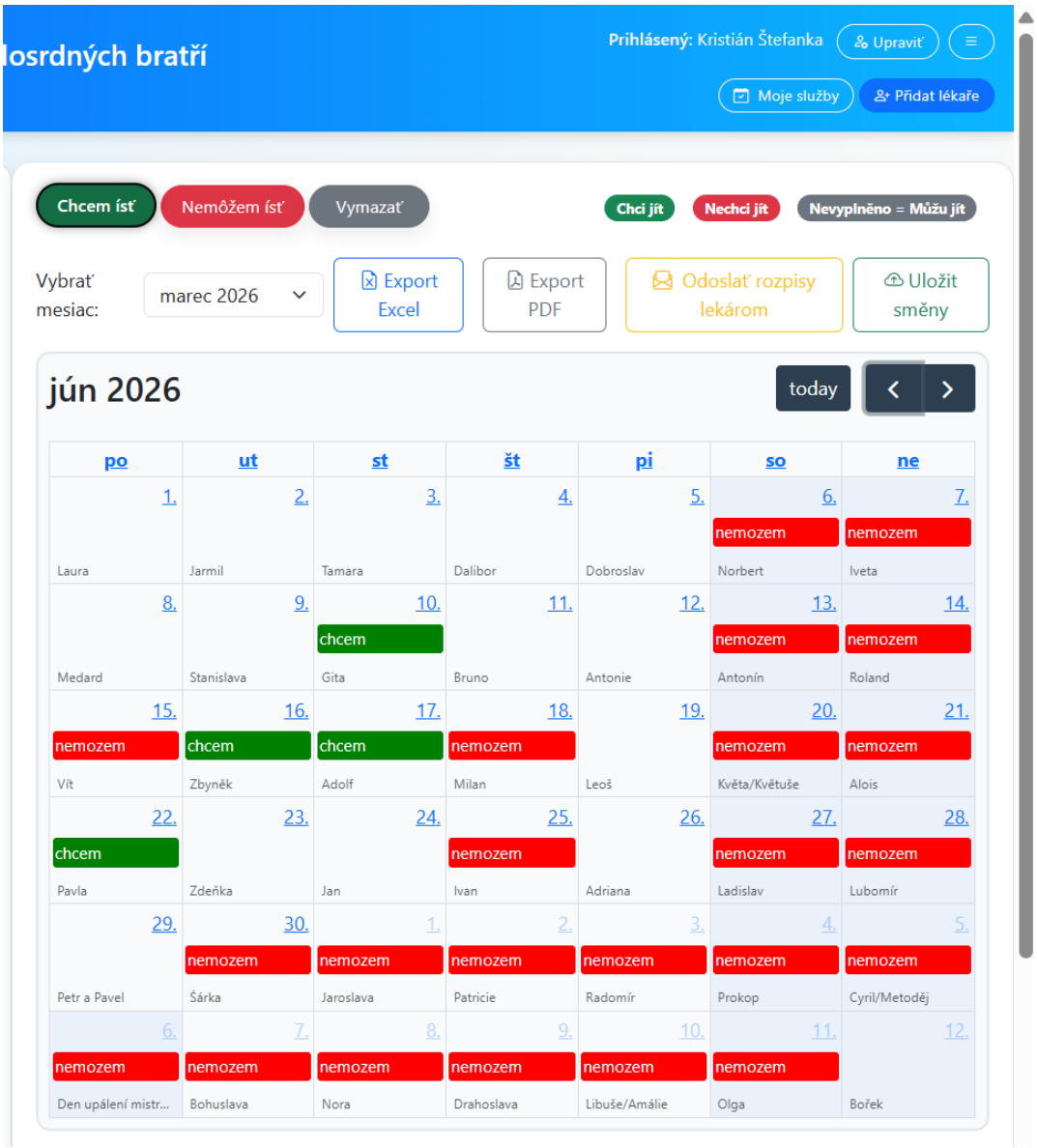

Fig.5. Interactive calendar for selecting doctor availability preferences.

The results showed a significant improvement compared to traditional Excel-based scheduling. In manual scenarios, the process of creating a schedule typically required several hours of work, especially when taking into account multiple constraints and preferences. In contrast, the proposed system was able to generate a complete schedule within a few seconds. This represents a substantial reduction in time and demonstrates the effectiveness of the automated approach. At the same time, the number of errors decreased significantly due to the implementation of automatic constraint validation, which prevents invalid assignments such as overlapping shifts or insufficient rest periods.

Another important aspect of the analysis was the evaluation of user experience. Feedback from users indicated that the system is intuitive and easy to use, even for individuals with limited technical experience. The visual representation of data through the interactive calendar interface was particularly appreciated, as it allows users to quickly understand and modify their preferences. The ability to input availability with a single click and immediately see the results of scheduling was identified as a major advantage over traditional methods, which often require manual updates and recalculations.

In addition to usability, the system was also evaluated in terms of stability and consistency. During testing, no critical failures or data inconsistencies were observed, which confirms the reliability of the implemented solution. The system maintained correct behavior even when multiple users interacted with it simultaneously, demonstrating its ability to handle concurrent access.

The system also demonstrated good scalability. Even with a higher number of users and increased complexity of scheduling requirements, the performance remained stable, and the scheduling process was completed within acceptable time limits. This confirms that the system is suitable not only for smaller departments but also for larger healthcare institutions where efficient scheduling is essential.

Furthermore, the evaluation included a comparison of qualitative aspects, such as transparency and fairness in shift allocation. The automated algorithm ensures that scheduling decisions are based on clearly defined rules and user preferences, which reduces the likelihood of subjective bias. This contributes to a more transparent and equitable distribution of shifts among doctors.

Overall, the analysis confirms that the proposed solution provides a significant improvement in efficiency, accuracy, and usability compared to traditional methods. The system not only reduces the time required for scheduling but also minimizes errors, improves user satisfaction, and provides a reliable and scalable platform for managing doctor shifts. These results highlight the potential of modern web-based applications to transform administrative processes in healthcare environments.

5 Identification of Problematic Areas

Despite the positive results, several limitations of the system were identified during testing and evaluation. One of the main limitations is the use of a heuristic algorithm, which does not guarantee an optimal solution. While the generated schedules satisfy all defined constraints and are suitable for practical use, it is possible that alternative solutions exist that could provide a more balanced distribution of shifts among doctors. This limitation is typical for heuristic approaches, which prioritize speed and simplicity over mathematical optimality. In more complex scenarios, especially with a higher number of constraints and users, the algorithm may produce solutions that are acceptable but not necessarily the most efficient or fair in the long term.

Another limitation is the lack of integration with external systems, such as hospital information systems, personnel management systems, or attendance tracking tools. In real-world healthcare environments, these systems are often interconnected, and the absence of integration may require redundant data entry or manual synchronization. Incorporating such integrations would significantly improve data consistency, reduce administrative workload, and enable more comprehensive management of hospital operations. For example, automatic synchronization with attendance systems could ensure that actual worked shifts are accurately recorded and reflected in future scheduling.

The current version of the system is primarily designed for web use, and although it is responsive and accessible on mobile devices through a browser, it does not include a dedicated mobile application. A native mobile application could provide a more optimized user experience, including faster interaction, offline access, and push notifications. Given that healthcare professionals often rely on mobile devices during their daily work, the absence of a mobile-specific solution may limit the system's usability in certain situations.

From a performance perspective, the system performs well under moderate load; however, its scalability has not been extensively tested in large-scale environments with hundreds of users and complex scheduling requirements. In such cases, performance optimization techniques, such as database indexing, caching mechanisms, or distributed system architecture, may be necessary to maintain efficiency and responsiveness.

Future improvements may also include the implementation of advanced optimization techniques, such as artificial intelligence or machine learning algorithms. These approaches could analyze historical scheduling data, identify patterns, and predict optimal staffing requirements. For example, machine learning models could learn from previous schedules and suggest improvements based on past conflicts, workload distribution, or user preferences. Such enhancements could significantly increase the quality of generated schedules and reduce the need for manual adjustments.

Additional features, such as notification systems or automated reminders, could further enhance the user experience and ensure better communication among staff members. Notifications could inform doctors about newly assigned shifts, changes in schedules, or upcoming duties, thereby reducing the risk of missed shifts or misunderstandings. Furthermore, implementing a feedback mechanism would allow users to report issues or suggest improvements, contributing to the continuous development of the system.

In conclusion, while the system provides a solid and functional foundation for efficient doctor shift scheduling, there are multiple opportunities for further development and improvement. Addressing the identified limitations and incorporating advanced features would enhance the system's robustness, scalability, and overall usability, making it more suitable for deployment in real-world healthcare environments.

Conclusion

The designed and implemented web-based system for doctor shift scheduling proved to be an effective solution for addressing the complex and time-consuming process of workforce planning in healthcare institutions. The system successfully integrates modern web technologies with a practical scheduling approach, providing a reliable platform that supports both administrators and medical staff. Particular emphasis was placed on improving efficiency, reducing human error, and ensuring fair distribution of shifts, which are key requirements in real-world hospital environments.

One of the main contributions of the system is the significant reduction in administrative workload. Traditional scheduling methods, especially those based on spreadsheets or manual planning, require considerable time and effort, often leading to inconsistencies and errors. In contrast, the proposed system automates the majority of scheduling tasks, allowing administrators to generate complete schedules within seconds. This not only increases productivity but also enables more frequent updates and adjustments in response to changing conditions.

The evaluation results confirmed that the system achieves a high level of accuracy and reliability. Automated constraint checking ensures that all generated schedules comply with predefined rules, such as rest periods, maximum workload, and availability constraints. As a result, the number of scheduling errors is significantly reduced compared to traditional methods. This contributes to safer and more efficient operation within healthcare institutions, where scheduling mistakes can have serious consequences.

Another important aspect of the system is its user-oriented design. The implementation of an interactive calendar interface allows doctors to easily input their availability preferences in a clear and intuitive manner. This not only simplifies user interaction but also increases engagement and satisfaction among staff members. The ability to visually interpret scheduling data and immediately see the results of changes represents a major improvement over static and less flexible traditional tools.

The system also demonstrated strong performance in terms of scalability and adaptability. It was able to handle different scheduling scenarios with varying numbers of users while maintaining stable performance and consistent results. This indicates that the solution is suitable not only for small departments but also for larger healthcare institutions with more complex requirements.

An additional benefit of the implemented system is the increased transparency and fairness in shift allocation. Since the scheduling process is based on clearly defined rules and user preferences, it minimizes the influence of subjective decision-making. This contributes to a more balanced distribution of workload among doctors and helps prevent conflicts within teams.

Despite these advantages, the system also highlights several areas for further improvement. The use of a heuristic algorithm, while efficient, does not guarantee an optimal solution in all cases. Future development could focus on incorporating advanced optimization methods, such as artificial intelligence or machine learning, to enhance decision-making and improve the overall quality of generated schedules. Furthermore, integration with external hospital systems and the development of a dedicated mobile application would significantly increase the system's practical usability.

In conclusion, the proposed system represents a modern and effective approach to doctor shift scheduling, addressing the limitations of traditional methods and demonstrating the potential of web-based solutions in healthcare management. The results confirm that digitalization and automation can significantly improve administrative processes, reduce errors, and enhance user experience. With further development and integration, the system has the potential to become a comprehensive tool for managing workforce scheduling in healthcare environments.

Acknowledgement

This work is part of the author's bachelor thesis, which is being prepared and will be defended in the current academic year.

I would like to express my sincere gratitude to my supervisor for his guidance, valuable advice, and continuous support throughout the development of this work.

I also thank the medical staff who participated in testing the system and provided useful feedback. The system was tested in the Hospital of the Merciful Brothers in Brno, where it is currently being gradually introduced into practical use, as no dedicated scheduling system was previously available.

References

[1] FullCalendar (2025). *JavaScript Event Calendar*.
Available at: https://fullcalendar.io/ (accessed January 10, 2026).

[2] PHP.net (2025). *PHP: Hypertext Preprocessor Documentation*.
Available at: https://www.php.net/docs.php (accessed January 10, 2026).

[3] Oracle (2025). *MySQL Documentation*.
Available at: https://dev.mysql.com/doc/ (accessed January 10, 2026).

[4] MDN Web Docs (2025). *JavaScript Guide*.
Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript (accessed January 10, 2026).

[5] Bootstrap (2025). *Bootstrap Documentation*.
Available at: https://getbootstrap.com/docs/ (accessed January 10, 2026).

[6] GeeksforGeeks (2025). *AJAX Introduction and Working*.
Available at: https://www.geeksforgeeks.org/ajax-introduction/ (accessed January 10, 2026).

[7] Nurse Scheduling Problem (2025). *Overview of scheduling in healthcare*.
Available at: https://en.wikipedia.org/wiki/Nurse_scheduling_problem (accessed January 10, 2026).

Authors

Kristián Štefanka
Faculty of Informatics
Pan-European University in Bratislava, Slovakia
xstefanka@paneurouni.com
Kristián Štefanka is a student of Applied Informatics at the Faculty of Informatics, Pan-European University in Bratislava. His academic interests include web application development, database systems, and optimization algorithms, particularly in healthcare process automation.

International Journal
Information Technology Applications (ITA)

Volume 15, Number 1, July 2026

SEMI-AUTOMATED QUANTITATIVE GAIT ANALYSIS USING VR MOTION TRACKING DATA AND POWER BI: A PILOT STUDY

Jakub Karol Brezina

Abstract:

This paper presents a pilot semi-automated framework for quantitative gait analysis based on motion data acquired from commercially available virtual reality tracking devices. The objective of the study is to evaluate the feasibility of using VR motion tracking technology for experimental quantitative assessment of human gait characteristics and to demonstrate a methodological workflow capable of transforming raw positional data into analytically interpretable outputs.

The proposed approach combines motion data acquisition using HTC Vive Tracker devices, signal preprocessing, Savitzky–Golay filtering, derivation of kinematic parameters, and semi-automated segmentation of characteristic gait-related motion segments. An important component of the framework is analytical visualisation and signal interpretation implemented in the Power BI environment, enabling efficient comparison of repetitive movement patterns and extraction of selected motion characteristics.

Pilot results indicate that commercially available VR tracking devices can provide sufficiently stable and interpretable motion data for experimental quantitative gait analysis. The proposed framework enables identification of characteristic gait segments, extraction of movement-related characteristics, and establishes a methodological basis for future comparative studies of motor behaviour. Although the presented approach does not represent a clinical diagnostic tool, it demonstrates a promising direction for further applications at the intersection of virtual reality, motion analytics, and movement assessment.

Keywords:

Virtual reality, gait analysis, motion tracking, kinematic analysis, signal processing, semi-automated movement assessment, Power BI.

Introduction

Objective assessment of human movement represents an important area of research with applications in biomechanics, rehabilitation, sports diagnostics, and potential future clinical support applications. Traditional gait assessment methods are often based on visual observation by trained specialists or on specialised motion capture systems that provide high measurement accuracy. However, such systems are frequently financially demanding, technically complex, and less accessible for broader experimental or practical deployment [1,2,5,7].

With the rapid development of virtual reality technologies, new opportunities have emerged for the use of commercially available motion tracking devices beyond their original purpose. Devices such as the HTC Vive Tracker enable continuous real-time tracking of spatial movement and provide motion tracking data that may serve not only for interaction within virtual environments, but also as a source of quantitative information for movement analysis [4,6].

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

Acquisition of motion data alone is insufficient for meaningful analytical interpretation. A major challenge lies in the transformation of raw positional measurements into interpretable movement characteristics. Motion tracking signals are commonly affected by measurement noise, tracker orientation variability, and short-term instability, while derivative-based calculations such as velocity and acceleration are particularly sensitive to signal imperfections. In addition, manual identification of individual movement events is time-consuming and may introduce inconsistencies caused by subjective interpretation [3,4,5].

For these reasons, there is an increasing need for semi-automated analytical approaches capable of improving the efficiency, consistency, and repeatability of motion data processing. The present study proposes a pilot semi-automated framework for quantitative gait analysis based on data acquired using VR motion tracking technology. The proposed workflow combines motion data acquisition, signal preprocessing, Savitzky–Golay filtering, derivation of kinematic parameters, semi-automated segmentation of characteristic gait-related motion segments, and analytical visualisation in the Power BI environment [5,7].

The main contribution of this work lies in demonstrating a practical and scalable analytical workflow capable of identifying repetitive gait patterns and extracting selected movement characteristics from commercially available VR tracking data. Although the presented approach is not intended as a clinical diagnostic system, it establishes a methodological basis for future comparative movement assessment and further research in objective motor behaviour analysis [1,5,7].

1 Materials and Methods

1.1 Data Acquisition

Experimental motion data used in this study were acquired using commercially available virtual reality motion tracking technology, specifically HTC Vive Tracker devices. These devices enable continuous real-time spatial tracking of a selected body segment and provide time-dependent positional coordinates suitable for subsequent movement analysis [4,6].

The experimental measurements focused on recording repetitive gait movement under controlled experimental conditions. Motion data were collected as sequential spatial coordinate measurements representing the movement trajectory of the tracked body segment during the experimental walking procedure. The pilot dataset consisted of motion recordings obtained from ten participants and was used for methodological framework verification to evaluate the feasibility of semi-automated gait analysis rather than clinical validation [1,5,7].

Particular attention was given to selecting the most analytically suitable coordinate axis for gait signal interpretation. Based on experimental observation, the vertical movement component was identified as the most stable and informative signal source. Compared to horizontal movement axes, the vertical coordinate demonstrated clearer periodic behaviour corresponding to repetitive gait cycles and was less affected by tracker orientation variability or individual participant positioning. This characteristic made it particularly suitable for subsequent segmentation and quantitative analysis [1,5,7].

The acquired raw motion data formed the input dataset for further preprocessing, filtering, and analytical transformation described in the following sections.

1.2 Signal Preprocessing

Raw motion tracking data acquired from the VR tracking system required preprocessing before meaningful analytical interpretation could be performed.

Motion signals generated by commercially available tracking devices inherently contain measurement noise, minor positional instability, and fluctuations caused by environmental factors or temporary variations in tracker orientation. These imperfections can significantly affect subsequent analytical calculations, particularly when derivative-based kinematic parameters are involved [3,4,5].

To improve signal quality while preserving essential movement characteristics, the acquired positional data were processed using the Savitzky–Golay filtering method. This approach was selected due to its ability to smooth noisy measurement data while maintaining local signal features, including peaks and periodic trends that are critical for identification of repetitive movement patterns. Compared to conventional smoothing techniques, the Savitzky–Golay filter provides improved preservation of signal structure, making it particularly suitable for movement-related time series analysis [3].

Following signal smoothing, the positional data were transformed into derived kinematic parameters. Velocity components (Vx, Vy, Vz) were calculated using first-order temporal differentiation between consecutive measurements, while acceleration components (Ax, Ay, Az) were subsequently derived from the velocity signals. These derived quantities provided a more detailed representation of dynamic movement behaviour and enabled clearer identification of transitions between characteristic gait-related motion segments. Since derivative calculations amplify even minor measurement noise, the preprocessing stage represented a critical step in ensuring analytical reliability [1,3,5].

The resulting processed signals formed the basis for the semi-automated segmentation and movement pattern analysis described in the subsequent section.

1.3 Semi-Automated Gait Segmentation

The identification of characteristic gait-related motion segments represents one of the key challenges in quantitative movement analysis. Traditional approaches often rely on manual inspection of movement signals, where individual gait cycles or relevant movement segments are identified by the analyst. Although such methods may be effective for small datasets, they are time-consuming, difficult to scale, and prone to inconsistencies resulting from subjective interpretation [1,2,7].

To address this limitation, a semi-automated segmentation approach was implemented within the proposed analytical framework. Instead of manually identifying individual movement events, the developed workflow enabled systematic detection of characteristic repetitive movement patterns directly from the processed motion signal. The segmentation process focused on identifying representative gait-related motion segments corresponding to distinct phases within the repetitive walking cycle [1,5,7].

The analytical segmentation was based on the periodic behaviour of the processed motion signal, particularly within the selected vertical movement component, which exhibited the clearest repetitive structure. By leveraging this signal regularity, the framework allowed efficient localisation of movement transitions and segmentation of gait-related motion events without requiring repeated manual intervention [1,5].

The semi-automated nature of the approach combines computational assistance with analyst supervision, ensuring both processing efficiency and interpretability of the results. This design significantly reduces analytical workload while maintaining sufficient control over movement interpretation, making the framework suitable for experimental comparative analysis and scalable processing of larger datasets [5,7].

1.4 Data Visualisation and Analytical Processing

Analytical processing and visual interpretation of the segmented motion data were performed using Microsoft Power BI as the primary analytical environment. The platform was selected due to its ability to efficiently transform structured datasets, support analytical workflows, and generate interpretable visual representations of time-dependent movement signals [5,7].

Within the proposed framework, Power BI was used not only for visualisation, but also as an analytical processing layer enabling transformation of raw motion tracking data into structured analytical outputs. Processed signals were organised, segmented, and evaluated in a manner that allowed identification of repetitive gait patterns and comparison of selected movement phases.

A key advantage of this approach lies in the integration of data transformation and visual analytics within a single environment. This significantly simplifies the analytical workflow compared to fragmented multi-tool processing approaches and supports repeatable evaluation of movement characteristics across multiple datasets. Visual representations of processed gait signals enabled efficient interpretation of periodic movement behaviour, identification of characteristic gait events, and qualitative comparison between segmented movement phases [5,7].

The resulting framework demonstrates that commercially available business intelligence tools can serve as effective analytical instruments for experimental motion analysis when combined with appropriate preprocessing and segmentation methodologies.

2 Results

The proposed semi-automated analytical framework was experimentally evaluated using motion data acquired during the pilot gait measurement procedure. Following preprocessing, filtering, and segmentation, the resulting signals exhibited clear and sufficiently stable periodic behaviour suitable for quantitative interpretation [4,5,7].

One of the primary findings of the study was that motion data acquired from commercially available VR tracking devices provided analytically usable information for experimental gait analysis. Despite the fact that the employed tracking technology was not originally designed as a specialised biomechanical motion capture system, the acquired signals demonstrated recognisable repetitive gait-related patterns that enabled systematic processing and interpretation.

The processed motion signal revealed clear characteristic periodic behaviour corresponding to repetitive walking movement. This periodicity provided the basis for subsequent segmentation and movement phase identification. The visual representation of the processed signal is shown in (Fig.1), where repetitive gait cycles can be clearly observed [1,5,7].

A second significant outcome of the study was the successful application of semi-automated gait segmentation. Based on the processed signal structure, characteristic gait-related motion segments could be systematically identified without requiring full manual annotation. This enabled clearer interpretation of the movement cycle and reduced analytical workload compared to purely manual segmentation approaches [1,2,7].

The segmentation output is illustrated in (Fig.2), where characteristic gait-related motion segments are visually distinguished within the analysed signal. The results demonstrate that repetitive motion events can be efficiently identified and analytically separated using the proposed framework.

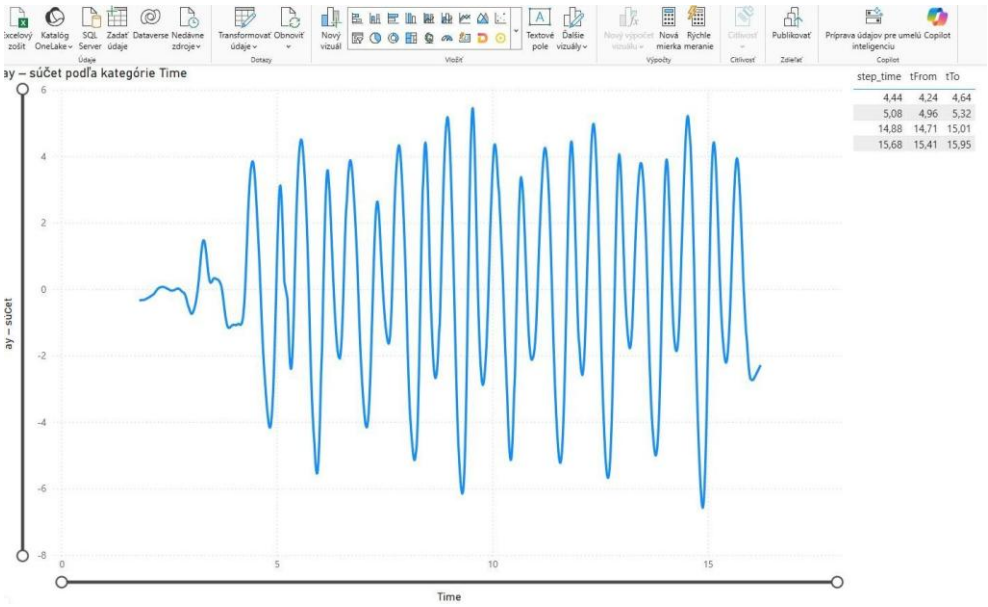

Fig.1. Processed periodic motion signal obtained from VR motion tracking during the pilot gait experiment.

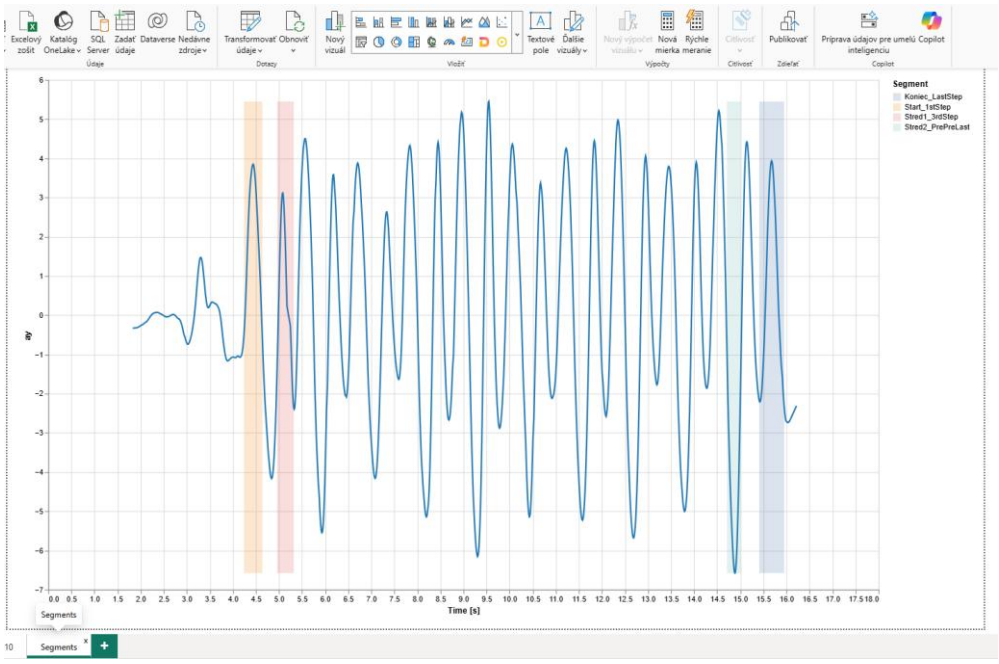

Fig.2. Semi-automated segmentation of characteristic gait-related motion events in the processed signal.

Overall, the experimental results indicate that the proposed workflow enables effective transformation of raw VR motion tracking data into structured analytical outputs suitable for experimental movement assessment. In addition to signal visualisation, the framework supports identification of repetitive gait-related motion segments and establishes a methodological foundation for future comparative quantitative analyses [5,7].

3 Discussion

The presented pilot study demonstrates that commercially available VR motion tracking technology can serve as viable source of analytically usable motion data for experimental quantitative gait analysis. Although such devices were originally developed for interaction within virtual environments rather than specialised biomechanical assessment, the obtained motion signals exhibited sufficient stability and periodic structure to support analytical processing and movement interpretation [4,5,7].

A major contribution of the proposed framework lies in the semi-automation of gait signal analysis. Manual identification of gait-related movement phases can be labour-intensive, inconsistent, and difficult to scale when processing larger datasets. By introducing a semi-automated segmentation workflow, the proposed approach reduces the analytical burden while improving consistency and repeatability of movement evaluation. This represents a practical advantage, particularly in experimental scenarios involving repeated measurements or comparative participant analysis [1,2,7].

An important methodological observation concerns the selection of the vertical movement component as the primary analytical signal source. Experimental evaluation indicated that the vertical axis provided clearer repetitive gait patterns and was less sensitive to tracker orientation variability compared to horizontal movement axes. This contributed significantly to segmentation reliability and overall interpretability of the motion signal [1,5,7].

The integration of preprocessing, segmentation, and visual analytics within the Power BI environment also proved to be a practical strength of the framework. Instead of relying on fragmented multi-tool analytical workflows, the proposed approach demonstrates that accessible business intelligence platforms can be effectively adapted for structured motion analysis tasks. This improves workflow efficiency and increases accessibility for future experimental applications [5,7].

However, several limitations must be acknowledged. The present work represents a pilot methodological study rather than a clinically validated diagnostic system. The framework was evaluated on a limited experimental dataset intended primarily for proof-of-concept verification. Consequently, the results should not be interpreted as evidence of diagnostic capability for specific pathological conditions [1,2,7].

Future work should focus on expanding the analysed dataset, incorporating a larger number of participants, and evaluating movement variability across healthy and potentially pathological gait patterns. Such extension would enable comparative quantitative analysis and support the development of more advanced movement assessment methodologies [1,2,7].

Overall, the findings suggest that semi-automated processing of VR motion tracking data repre-sents a promising direction for future research at the intersection of virtual reality technologies, motion analytics, and quantitative human movement assessment [4,5,7].

Conclusion

This paper presented a pilot semi-automated analytical framework for quantitative gait analysis based on motion data acquired from commercially available VR tracking devices.

The proposed methodology demonstrated that technologies originally intended for virtual reality interaction can also provide a practical and analytically usable source of motion data for experimental movement assessment [4,5,7].

The main contribution of the study lies in the development of a structured analytical workflow combining motion data acquisition, signal preprocessing, Savitzky–Golay filtering, derivation of kinematic parameters, semi-automated gait segmentation, and analytical visualisation using Power BI. This integrated approach enables more efficient and repeatable movement analysis compared to purely manual processing methods [1,3,4,5,7].

The experimental results confirmed that the acquired motion signals contained sufficiently stable and interpretable periodic patterns corresponding to repetitive gait movement. These characteris-tics enabled identification of representative gait-related motion segments and demonstrated the feasibility of transforming raw VR tracking data into meaningful analytical outputs [1,5,7].

Although the proposed framework does not represent a clinical diagnostic tool, it establishes a methodological basis for future comparative movement analysis. Further development may include larger participant datasets, comparative evaluation of healthy and pathological gait behaviour, and extension toward more advanced quantitative movement assessment applications [1,2,7].

The presented work demonstrates a promising interdisciplinary research direction connecting virtual reality technologies, data analytics, and quantitative human movement analysis [4,5,7].

Acknowledgement

The author gratefully acknowledges doc. RNDr. Eugen Ružický, CSc. for his valuable guidance, expert consultation, and constructive feedback throughout the development of this study. His methodological expertise and professional support significantly contributed to the direction and overall quality of the presented research.

References

[1] Whittle, M. W. (2007). Gait Analysis: An Introduction.
4th ed. Butterworth-Heinemann, Oxford, UK.

[2] Baker, R. (2013). Measuring Walking: A Handbook of Clinical Gait Analysis.
Mac Keith Press, London, UK.

[3] Savitzky, A., Golay, M. J. E. (1964). Smoothing and differentiation of data
by simplified least squares procedures.
Analytical Chemistry, vol. 36, no. 8, pp. 1627–1639. doi: 10.1021/ac60214a047

[4] Niehorster, D. C., Li, L., Lappe, M. (2017). The accuracy and precision of position
and orientation tracking in the HTC Vive virtual reality system for scientific research.
i-Perception, vol. 8, no. 3, pp. 1–23. doi: 10.1177/2041669517708205

[5] Tao, W., Liu, T., Zheng, R., Feng, H. (2012). Gait analysis using wearable sensors.
Sensors, vol. 12, no. 2, pp. 2255–2283. doi: 10.3390/s120202255

[6] Jerald, J. (2015). The VR Book: Human-Centered Design for Virtual Reality.
Association for Computing Machinery, New York, USA.

[7] Muro-de-la-Herran, A., Garcia-Zapirain, B., Mendez-Zorrilla, A. (2014). Gait analysis methods:
An over-view of wearable and non-wearable systems, highlighting clinical applications.
Sensors, vol. 14, no. 2, pp. 3362–3394. doi: 10.3390/s140203362

Author

Jakub Karol Brezina
Faculty of Informatics
Pan-European University in Bratislava, Slovakia
jakubkarolbrezina@gmail.com
Jakub Karol Brezina is a third-year undergraduate student of Applied Informatics at Pan-European University. His academic interests include virtual reality technologies, data analytics, human movement analysis, and digital system design. His current research focuses on the application of VR motion tracking technologies for quantitative movement analysis and experimental data interpretation. He is particularly interested in interdisciplinary applications connecting information technologies, data analytics, and human movement assessment.

International Journal
Information Technology Applications (ITA)

Volume 15, Number 1, July 2026

ANALYSIS OF QUIZZES FOR THE PRINCE2 METHODOLOGY IN RELATION TO THE ASSESSMENT OF STUDENT KNOWLEDGE AND THE AUTOMATED GRADING OF THE PROJECT MANAGEMENT COURSE

Natália Gonosová

Abstract:

This study examines the use of quizzes as a tool for assessing students' knowledge in a university course on project management focused on the PRINCE2 methodology. The study is based on a selected set of 80 multiple-choice questions, from which three tests of 20 questions each were selected and compiled; these were semi-automatically implemented in Microsoft Excel and subsequently in Google Forms to support scalable delivery, automatic grading, and structured data collection. Using descriptive statistics and item-level error analysis, the results compare performance on midterm exams and during the final exam period. The findings show a significant improvement from an average success rate of 68.72% on midterm exams to 92.89% during the exam period, suggesting that more intensive preparation is associated with better results, although claims about relationships remain weaker due to the still low granularity of the questions. At the item level, several questions showed consistently low success rates (approximately 45–64%), particularly regarding PRINCE2 management and decision-making responsibility, suggesting systematic conceptual difficulties. Automated analysis (Excel/Google Forms) provides a framework for rapid feedback to students and practical diagnostics for instructors, thereby supporting targeted improvements in teaching and assignment design.

Keywords:

PRINCE2, project management education, knowledge assessment, quiz analysis, automated grading, Microsoft Excel, Google Forms.

Introduction

Education in project management today lies between two dominant approaches—traditional (process-based) methodologies and agile frameworks. In the academic environment, there is a growing need not only to provide students with theoretical knowledge, but also to objectively and effectively assess their understanding of key concepts. One methodology that offers a clear structure of roles, decision points, and management processes is PRINCE2 [1], [2]. Its process-oriented nature, emphasis on responsibilities, and control mechanisms create a solid foundation for the systematic development of students' project thinking.

From a didactic perspective, PRINCE2 primarily supports students' ability to understand the logic of project management: who makes decisions, when decisions are made, and what information is required to make them. This type of "structured" understanding is important not only for mastering terminology, but also for navigating organizational roles and correctly interpreting project situations. On the other hand, agile frameworks such as Scrum promote flexibility, teamwork, and adaptability to change, but often do not provide such an explicit and formally defined management structure.

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

From the perspective of knowledge assessment, PRINCE2 therefore offers an advantage—its concepts are more clearly mappable to test items and allow for more precise diagnosis of areas where students struggle.

The core of the work lies in identifying questions that most reliably distinguish different levels of student understanding, as well as in detecting problematic topics (e.g., roles, processes, and decision-making responsibilities) that require targeted instructional improvement. The contribution of this paper is the integration of a didactic perspective with analytical processing of test results, enabling the creation of a repeatable and objective assessment mechanism that provides immediate feedback for both students and instructors.

The paper addresses the following research questions:

1. **RQ1:** What is the overall success rate of students in PRINCE2 tests during the semester and during the examination period, and what are the differences between them?
2. **RQ2:** Which questions (or PRINCE2 topic areas) achieve the lowest success rates and repeatedly appear as problematic?
3. **RQ3:** How does student performance change between semester and examination testing for questions of varying difficulty (easy/medium/hard)?
4. **RQ4:** To what extent can automated processing of results (e.g., in Excel or Google Forms) be used for fast and consistent test evaluation and reporting for instructors?
5. **RQ5:** What recommendations for improving teaching and test design emerge from the analysis (e.g., the need for visualizations, examples, or explanations of incorrect answers)?

At the beginning of the question-development process, we formulated two hypotheses that we wanted to test:

H1: Students' average scores on PRINCE2 tests during the exam period are higher than their average scores during the semester.
H2: The improvement in scores between the semester and the exam period is more pronounced in questions grouped by topic area.

1 Methodology

1.1 Test Design

The testing process was designed with the aim of comprehensively assessing the level of students' knowledge while also creating conditions for its detailed analysis. Emphasis was placed not only on the efficient implementation of testing, but also on obtaining accurate and informative results that enable objective evaluation and the identification of problematic areas within the subject matter.

In the initial phase, a database of 80 test questions was created, covering the full scope of the curriculum and various levels of difficulty. After revision, 60 questions were selected for testing, while 20 were excluded due to their overly theoretical nature. From the selected questions, three separate tests were subsequently constructed, each consisting of 20 questions.

The questions were designed as multiple-choice items with four possible answers (A, B, C, D), with only one correct answer in each case. This approach ensured unambiguous evaluation, eliminated subjectivity, and enabled straightforward automated processing of results.

During the development of the questions, particular emphasis was placed on clarity of formulation and relevance of content. The questions were based on validated study materials and covered key thematic areas of the course. At the same time, an appropriate level of difficulty was

considered, with a combination of easier and more challenging questions allowing for better differentiation among students and providing a realistic picture of their level of knowledge.

1.2 Implementation of Tests

After designing the test questions, the next phase was their practical implementation, which was carried out in two steps – first in the Microsoft Excel environment and subsequently on the online platform Google Forms [4]. This two-step approach made it possible to effectively combine the advantages of local data processing with the capabilities of online testing and response collection.

In the first phase, the tests were implemented in Microsoft Excel, which served as the main tool for design, functionality verification, and ongoing evaluation of the tests (Fig.1). Each test was created as a separate file, and its structure consisted of multiple worksheets. The main worksheet was used for entering students' answers.

PRINCE2 Foundation – Interactive Test (a–d)

Instructions: In the columns, select one answer from the drop-down list. The correctness is evaluated automatically. Results can be found in the 'Overview' sheet.

#	Question	a)	b)	c)	d)	Your Answer	Correctness
6	Which of the following is NOT one of the 7 PRINCE2 processes?	a) Directing a Project	b) Closing a Project	c) Planning a Project	d) Controlling a Stage	a	Incorrect
7	Who is responsible for approving the Business Case?	a) Project Manager	b) Project Board (Project Board)	c) Team Manager	d) Project Sponsor	b	Correct
8	Which document defines requirements for the quality of a specific product?	a) Product Description	b) Quality Register	c) Risk Register	d) Issue Register	a	Correct
9	What is the PRINCE2 principle that deals with learning from previous projects?	a) Managing by Stages	b) Learning from Experience	c) Focus on Products	d) Managing by Exception	c	Incorrect
10	Which process closes a project in PRINCE2?	a) Starting up a Project	b) Initiating a Project	c) Managing a Stage Boundary	d) Directing a Project	a	Correct
11	Which document identifies and records risks?	a) Issue Register	b) Risk Register	c) Quality Register	d) Product Description	b	Correct
12	Who is responsible in PRINCE2 for delivering work packages?	a) Project Manager	b) Team Manager	c) Project Management Team	d) Project Sponsor	b	Correct
13	What is the purpose of a PRINCE2 Theme?	a) Managing Stakeholders	b) Configuration	c) Organization	d) Communication	c	Correct
14	Which process ensures that the project is successfully concluded?	a) Closing a Project	b) Directing a Project	c) Managing Product Delivery	d) Initiating a Project	a	Correct
15	Which document records issue resolution during a project?	a) Risk Register	b) Issue Register	c) Quality Register	d) Business Case	b	Correct
16	Which PRINCE2 principle is about tailoring the method to the project context?	a) Managing by Stages	b) Tailoring to the Project Environment	c) Focus on Products	d) Managing by Exception	c	Incorrect
17	What is the main output of the Initiating a Project process?	a) Project Brief	b) Project Initiation Document (PID)	c) Business Case	d) Product Description	b	Correct
18	Who is responsible for managing (controlling) a stage?	a) Project Manager	b) Project Management Team	c) Team Manager	d) Project Sponsor	a	Correct
19	Which plan typically includes the delivery of work packages to a team level?	a) Stage Plan	b) Team Plan	c) Project Plan	d) Exception Plan	a	Incorrect
20	Which process performs end of stage control as part of the stage?	a) Controlling a Stage	b) Managing Product Delivery	c) Directing a Project	d) Closing a Project	a	Correct

Fig.1. Test sheet (a test worksheet for answering PRINCE2 questions with automatic correctness evaluation).

An integral part of the implementation in Excel was the automatic evaluation of answers, which was carried out using logical functions, primarily the IF function. These functions compared the answers entered by students with the correct answers stored in the system. This evaluation method ensured speed, accuracy, and objectivity without the need for manual intervention.

After verifying the functionality of the tests in the Microsoft Excel environment, the tests were transferred to the Google Forms platform (Fig.2), which enabled easier distribution among students and more efficient data collection. The tests were made available via a link, increasing their accessibility.

From the original three tests, six variants were created, each containing a different combination of questions from the database. These variants were used during different testing sessions, contributing to the systematic organization of testing and an even distribution of difficulty.

The Google Forms platform also provided tools for automated evaluation, particularly the ability to define correct answers and set a random order of questions. The system then automatically evaluates responses and assigns points, eliminating the need for manual grading, reducing the risk of errors, and increasing the overall efficiency of the process (Fig.3), (Fig.4).

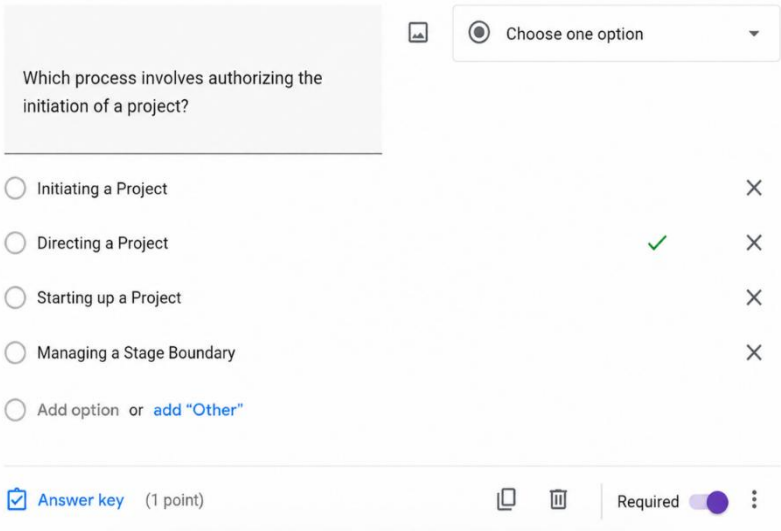

Fig.2. Google Forms – Project management quiz (creating a question in Google Forms).

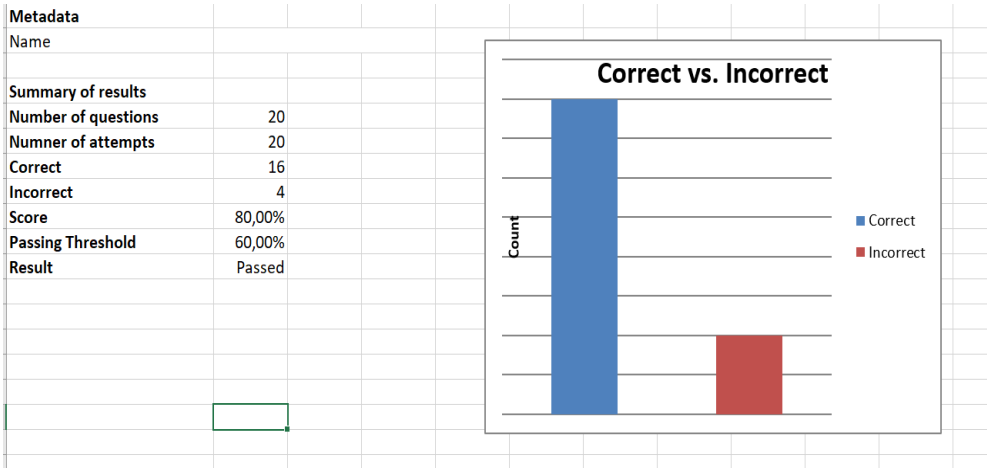

Fig.3. Test Results Summary and Performance Chart
(summary of test results with score and correct vs. incorrect comparison).

Another significant advantage was the possibility of immediate feedback for students. After submitting the test, students were able to instantly view their results. Google Forms also ensured centralized data collection, with all responses automatically stored in a structured spreadsheet. This spreadsheet contained information about individual students' answers, their final scores, and the time of test submission. The data could subsequently be exported to the Microsoft Excel environment, where it was further processed and analyzed.

Časová pečiatka	Výsledok	Piezvisko	Ktorá téma PRINCE2 zabezpečuje, že projekt	Kto je zodpovedný za Business Case a je jedi v	Ktorý proces udeľuje povol
10.12.2025 15:24:12	4 / 10	-	Business Case	Project Manager	Directing a Project
10.12.2025 15:25:00	10 / 10	-	Business Case	Executive	Directing a Project
10.12.2025 15:31:13	10 / 10	-	Business Case	Executive	Directing a Project
17.12.2025 11:26:14	7 / 10	-	Pokrok	Executive	Starting up a Project
17.12.2025 11:26:17	6 / 10	-	Organizácia	Senior User	Initiating a Project
10.1.2026 10:50:52	8 / 10	-	Business Case	Executive	Initiating a Project
10.1.2026 10:57:53	7 / 10	-	Business Case	Project Manager	Starting up a Project
14.1.2026 10:55:00	10 / 10	-	Business Case	Executive	Directing a Project
17.1.2026 10:46:08	10 / 10	-	Business Case	Executive	Directing a Project
17.1.2026 11:17:54	10 / 10	-	Business Case	Executive	Directing a Project

Fig.4. Viewing quiz responses in Google Forms
(Google Forms quiz responses with scores and answers:
Timestamp, Result, Name (empty), Which topic, Responsible, Supervisor)

2 Analysis

After the completion of testing, the collected data were processed in the Microsoft Excel environment, which served as the main tool for their analysis and interpretation. This phase represented a key step of the entire process, as it enabled the evaluation of student performance, assessment of test quality, and identification of problematic areas of the subject matter.

In the first step, the data were collected, organized, and prepared for further processing. Subsequently, basic statistical indicators were calculated to provide an overview of the testing process. These indicators mainly included the number of correct and incorrect answers, the total number of participating students, and the average success rate of the test.

In addition to the overall success rate, special attention was paid to the analysis of individual questions, which made it possible to identify both easier questions and those with a higher level of difficulty. Based on the achieved success rate, the questions were subsequently divided into three categories – easy, medium, and difficult. This classification was based on predefined success rate intervals, which allowed for the systematic assignment of each question to the appropriate category.

For each test, charts were created showing the number of correct answers (Fig.5), (Fig.6), which made it possible to quickly identify differences in student performance and also highlighted questions that were either easier or more difficult.

The analysis also included a comparison of results between testing conducted during the semester and the examination period, which made it possible to track the development of students' knowledge and assess the effectiveness of teaching. Data processing was carried out systematically in the Microsoft Excel environment as an analytical pipeline, which included steps from data import and cleaning, through the calculation of statistical indicators, to visualization and interpretation of results. The advantage of this approach lies in its systematic nature and repeatability, as the same procedure can be applied to future testing, thereby reducing the risk of errors and increasing the reliability of the results.

No.	Question	Correct answer	Correct	Incorrect	Number of students	Success rate	Question difficulty
1	What is the main output of the project start-up phase (Initiating a Project)?	Authorization from Directing a Project	5	6	11	45.45%	Difficult
2	Which process authorizes the initiation of a project?	Directing a Project	10	6	16	62.50%	Difficult
3	Which role ensures that the project delivers the required outputs and outcomes?	Senior User	7	4	11	63.64%	Difficult
4	Which of the following is NOT one of the six quality management objectives in PRINCE2?	Resource capacities	11	5	16	68.75%	Difficult
5	Who is responsible for continuously ensuring and improving the quality of the supplied products?	Senior Supplier	5	2	7	71.43%	Difficult
6	Who is responsible for managing (controlling) a stage?	Project Manager	8	2	10	80.00%	Medium
7	Who is responsible in the Business Case at a single point of accountability for the project?	Executive	13	3	16	81.25%	Medium
8	When is the Lessons Report created?	At the end of the project	9	2	11	81.82%	Medium
9	Which process produces the Project Brief?	Starting up a Project	6	1	7	85.71%	Medium
10	Which document specifies the quality details of the product within the quality tolerances?	Product Description	6	1	7	85.71%	Medium

Fig.5. Quiz results analysis (overview table showing quiz statistics, correct answers, success rates, and question difficulty).

Fig.6. Question error rate chart (bar chart displaying the number of incorrect answers for individual quiz questions).

3 Results

3.1 Results During the Semester

During the semester, three continuous tests were conducted with the aim of monitoring students' level of knowledge and identifying possible gaps in their understanding of the subject matter. The achieved success rates of the individual tests were 70.17%, 69.83%, and 66.17%, with an average success rate of 68.72%. This result provides an overall picture of students' knowledge during the teaching period and serves as a basis for comparison with the results from the examination period (Fig.7).

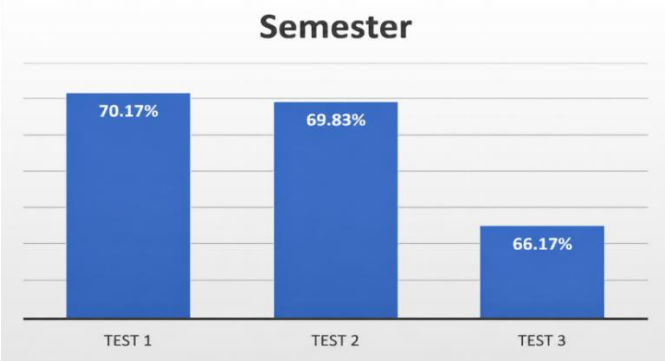

Fig.7. Semester test results (bar chart comparing success rates across three semester tests).

From a developmental perspective, a relatively stable level of success can be observed, with a slightly decreasing trend. This decline may be related to the increasing difficulty of the subject matter and the accumulation of knowledge, which place higher demands on the ability to connect and apply concepts. The results of continuous testing serve an important diagnostic function, as they allow for the identification of weaker areas and provide feedback for adjusting the teaching process.

3.2 Results During the Examination Period

During the examination period, six tests were conducted to assess students' level of knowledge after completing the course and preparing for the exam. The achieved success rates of the individual tests were 96%, 85%, 86.36%, 97.14%, 98.57%, and 94.29%. The average success rate reached 92.89%, which represents a significant improvement compared to continuous testing.

Most of the tests showed very high success rates, with several exceeding 95%, indicating a strong understanding of the subject matter. However, slight differences were observed between individual tests. The lowest success rate was recorded in the second test (85%), likely due to higher difficulty or the presence of more challenging questions. In contrast, the highest success rate was achieved in the fifth test (98.57%), indicating excellent mastery of the given topic. Overall, the results during the examination period do not show a significant upward or downward trend, but rather slight fluctuations around a high level of success.

3.3 Comparison of Results

Based on the processed data, a comparison was made between the results achieved during the semester (Fig.8) and those from the examination period. The average success rate during the semester was 68.72%, while during the examination period it reached 92.89%, representing an increase

of more than 24 percentage points (Fig.9). This difference indicates a significant improvement in students' level of knowledge.

The results indicate a significant shift in performance between continuous testing and final assessment. While continuous tests provide an overview of the current level of knowledge and help identify weaker areas, the results from the examination period reflect a more comprehensive understanding of the subject matter after revision and consolidation.

The improvement can be explained mainly by more intensive student preparation and the cumulative effect of learning, as students have more time during the examination period to systematize their knowledge. Continuous testing also serves a formative function – it provides feedback and guides students in their preparation for the exam.

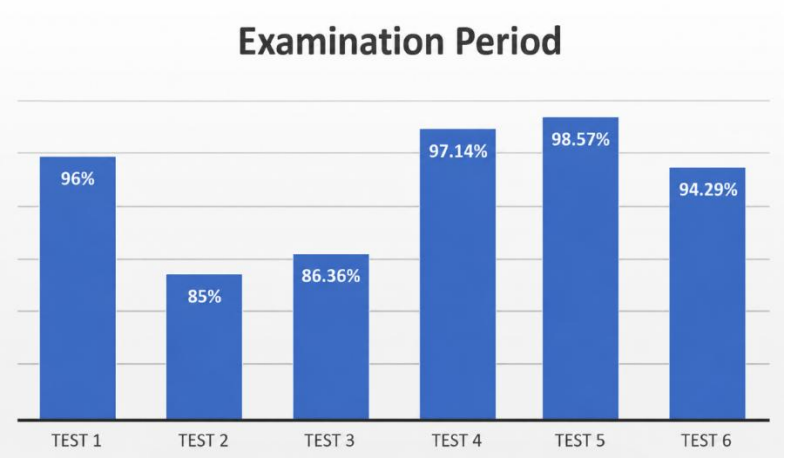

Fig.8. Examination period test results
(bar chart showing success rates across six tests during the examination period).

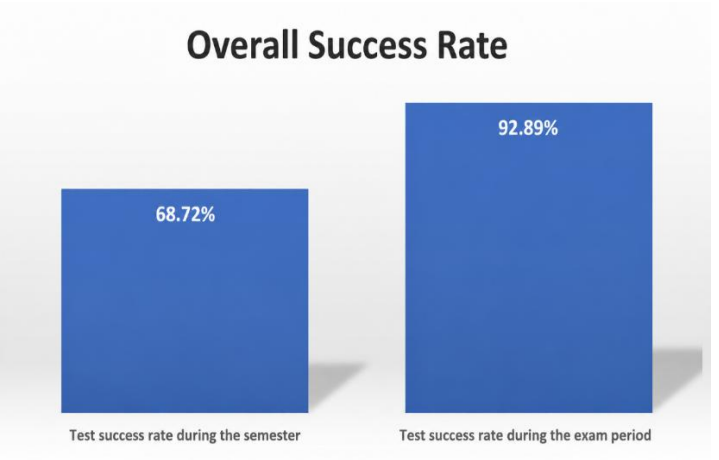

Fig.9. Overall success rate comparison
(Bar chart comparing test success rates during the semester and examination period.)

3.4 Hypothesis Testing H1

Hypothesis H1 assumes that student performance during the examination period is higher than during the semester. Its objective was to verify whether there is an improvement in knowledge as a result of more intensive preparation and revision of the subject matter.

The average success rate during the semester was 68.72%, while during the examination period it reached 92.89%, representing an increase of more than 24 percentage points (Fig.9). This difference indicates a significant improvement in student performance.

The improvement can be explained mainly by more intensive preparation, systematic revision of the subject matter, and greater time available for studying. An important role is also played by the experience gained during the semester, as continuous testing helps students identify weaker areas and better prepare for the final assessment.

Based on these findings, it can be concluded that hypothesis H1 was confirmed, as students achieved significantly higher success rates during the examination period.

3.5 Hypothesis Testing H2

To verify hypothesis H2, which assumes a more significant improvement in performance on difficult questions during the examination period, a detailed analysis at the level of individual questions is required. This primarily involves comparing their success rates during the semester and the examination period within individual difficulty categories.

Such an analysis requires the creation of a per-question success matrix, which would contain data on student performance for each question in both periods. Based on this, it would be possible to accurately assess changes, especially for difficult questions.

However, based on the available aggregated data, it is not possible to conclusively confirm or reject hypothesis H2. The results suggest that some questions remain problematic even during the examination period.

From a methodological perspective, this analysis can be automated in the Microsoft Excel environment, for example using VBA [3], which enables efficient data processing and more accurate comparison of results.

3.6 Discussion

The significant increase in student success rates during the examination period can be explained by a combination of several factors. A key role is played mainly by higher motivation and more intensive preparation, which includes systematic revision and reinforcement of knowledge. An important factor is also the formative nature of continuous testing during the semester, which provides feedback, helps identify weaker areas, and develops students' test-taking skills. The combination of these factors leads to a gradual improvement in performance, which becomes especially evident during examinations.

An important contribution of this work is also the use of automation in data processing through Microsoft Excel and VBA. This approach enables repeatable and consistent processing of results, minimizes the risk of errors, and significantly increases the efficiency of the analytical process. The automated procedure includes data import and cleaning, calculation of statistical indicators, analysis of question success rates, and creation of graphical outputs, allowing the rapid generation of clear and structured reports.

Another important aspect of the teaching process was the connection of theory with practice through semester projects implemented in the Microsoft Project environment. Students worked on projects from technical fields, where they applied principles of planning, resource management, and scheduling. In this context, traditional methodologies such as PRINCE2 proved to provide an appropriate management structure, while agile approaches were less applicable in their pure form.

3.7 Problematic Questions and Suggestions for Improvement

Based on the analysis of test results, problematic questions with the lowest success rates were identified. These were mainly questions 1 to 3, whose success rates ranged approximately from 45% to 64% (Fig.10). These questions were also classified as difficult, confirming their higher level of complexity for students.

Number	Question	Correct Answer	Correct	Incorrect	Number of Students	Success Rate	Difficulty Level
1	What is the first step in initiating a project (Initiating a Project)?	Authorization from the Directing a Project	5	6	11	45.45%	Difficult
2	Which process is responsible for initiating a project?	Directing a Project	10	6	16	62.50%	Difficult
3	Who is responsible for delivering operational outputs and benefits of the project?	Senior User	7	4	11	63.64%	Difficult
4	Which of the following is NOT one of the seven performance targets in PRINCE2?	Resource capacity	11	5	16	68.75%	Difficult
5	Who is responsible for the continuous supply of products and ensuring their quality?	Senior Supplier	5	2	7	71.43%	Difficult

Fig.10. Analysis of difficult quiz questions (table showing the least successful quiz questions, including correct answers, success rates, and difficulty levels).

The content of these questions focused mainly on the Directing a Project process, the Senior User role, and the fundamental principles of project roles within the PRINCE2 methodology. The lower success rate indicates insufficient understanding of these areas, suggesting that the problem is not random but systematic.

The main cause lies primarily in the theoretical nature of the questions, which requires precise knowledge of terminology and the ability to correctly interpret individual concepts in context. Students particularly struggled with understanding the strategic role of the Directing a Project process and with correctly distinguishing project roles, especially the Senior User role, which they often confused with other project roles.

To improve teaching effectiveness, it is recommended to use visual aids such as process diagrams and organizational charts, which help students better understand the relationships between the individual elements of the methodology. Equally important is connecting theory with practical examples and model situations that allow better application of knowledge.

At the same time, greater attention should be devoted to explaining incorrect answers and including more practice tasks focused on the identified problematic areas. These measures can contribute to deeper understanding and improved student performance.

3.8 Limitations and Constraints

When interpreting the obtained results, it is necessary to consider several limitations that may have affected the accuracy and completeness of the analysis.

One of the main limitations is the unavailability of detailed data at the level of individual questions for both observed periods. To accurately verify hypothesis H2, it would be necessary to work with a per-question success matrix that would allow comparison of student performance on

each question during both the semester and the examination period. Due to the absence of such data, the interpretation of the results can only be considered partial.

Another factor that may have influenced the results is the different composition of students across individual testing periods. Different groups of students may have varying levels of knowledge, motivation, or experience, which may be reflected in the overall success rate.

The results may also have been affected by differences in the difficulty of individual test variants. Although creating multiple test variants helps reduce the possibility of cheating, it may also lead to slight differences in difficulty that can influence the achieved results.

Based on these limitations, the results should be interpreted with a certain degree of caution. Nevertheless, the obtained data provide a relevant insight into the level of student knowledge and make it possible to identify the main trends and problematic areas.

Conclusion

In this paper, we analyzed the PRINCE2 test questions used in a project management course with the aim of:

(i) comparing students' performance on formative assessments during the semester with their performance on assessments administered during the exam period,
(ii) identify tasks with consistently low success rates, and
(iii) evaluate the benefits of automated data collection, scoring, and reporting of results.

The findings revealed a significant difference between these two periods: the average success rate was 68.72% during the semester and 92.89% during the exam period, representing an increase of more than 24 percentage points. This empirical pattern is consistent with the interpretation that formative assessment can serve as a formal function and, when combined with more intensive preparation during the exam period, is associated with higher success rates; however, given the study design and data availability, causal explanations should be approached with caution.

Some questions were identified as consistently problematic, exhibiting the lowest success rates in relation to certain PRINCE2 processes and principles. From a pedagogical perspective, these results highlight the need to strengthen instruction in the identified areas and revise the wording of items to promote a logical understanding of managerial responsibilities and process interconnections, rather than primarily testing memorization of terminology. Regarding data processing, automated grading in Excel/Google Forms increased the repeatability and consistency of scoring and enabled the rapid generation of analytical outputs and timely feedback for students.

The interpretation of results is limited by differences in the difficulty of individual test variants. Therefore, a priority for further development is to implement systematic item-level data collection throughout the semester, define metrics (including linking items to subject areas and difficulty levels), and standardize the creation of test variants in terms of both content and difficulty. In conclusion, it can be stated that the combination of structured PRINCE2 concepts with regular assessment and automated processing provides a functional framework for a more objective evaluation of knowledge and for the continuous improvement of teaching quality in the Project Management course.

Acknowledgement

I would like to express my sincere gratitude to my supervisor Eugen Ružický for guidance, support, and valuable advice during the preparation of this work. I also thank all students who participated in the testing process and contributed to the data collection used in this analysis. Special thanks belong to the Faculty of Informatics Pan-European University for providing the necessary resources and support throughout the project.

References

[1] PRINCE2 Wiki. (2025). PRINCE2 Wiki – Project Management Methodology. Retrieved online, May 6, 2025, from https://prince2.wiki/

[2] PeopleCert. (2025). What Is PRINCE2? The Definition, History & Benefits. Retrieved online, May 6, 2025, from https://www.prince2.com/eur/what-is-prince2

[3] Microsoft Corporation. (2025). Reference Object Library Reference for Office – VBA Documentation. Retrieved online, May 7, 2025, from https://learn.microsoft.com/sk-sk/office/vba/api/overview/library-reference/reference-object-library-reference-for-office

[4] Google LLC. (2025). Create Quizzes in Google Forms. Retrieved online, May 7, 2025, from https://support.google.com/docs/answer/6281888?hl=en&co=GENIE.Platform%3DAndroid&sjid=9304705905418317780-EU

Author

Natália Gonosová
Faculty of Informatics
Pan-European University in Bratislava, Slovakia
xgonosova@paneurouni.com

Natália Gonosová is a student of Applied Informatics at the Faculty of Informatics, Pan-European University in Bratislava, Slovakia. My academic interests include project management methodologies, especially PRINCE2, and their practical application in process management and optimization.

International Journal
Information Technology Applications (ITA)

Volume 15, Number 1, July 2026

AI-BASED DEEP FAKE AVATARS USING GAUSSIAN SPLATTING

Ondrej Hudcovič

Abstract:

This paper investigates the technical feasibility of creating an interactive conversational avatar using Gaussian Splatting technology in the Unreal Engine environment. The study focuses on verifying the possibility of implementing a talking avatar without using parametric face models. The result of the study is a prototype consisting of a local FastAPI backend and a client application in Unreal Engine 5.3.2, which integrates speech recognition, response generation by a local large language model, speech synthesis and data generation for lip synchronization. The avatar is rendered using the 3D Gaussian Splatting plugin from Luma AI. The paper includes the implementation of the application in the VR environment of Unreal Engine. The implementation section of the study also introduces comparative testing of four large language models and latency measurement of the entire conversational data pipeline. The study confirms the technical feasibility of the combination of Gaussian Splatting and a conversational AI application only partially, as the main open problem remains the implementation of the avatar's lip animation.

Keywords:

Gaussian Splatting, conversational avatar, virtual reality, large language models, Unreal Engine.

Introduction

Virtual reality applications have become increasingly popular in recent years, not only in the fields of entertainment and gaming, but also in the fields of tourism, education, healthcare, schooling, remote communication and many others. One way to achieve high-quality and immersive VR experiences is to have avatars with which users can interact naturally. Traditional approaches to creating avatars involve time-consuming manual modeling or the use of expensive motion capture systems. This naturally poses a problem for small developers and teams of people with limited resources.

Recent advances in neural rendering, especially Gaussian Splatting, open up new possibilities for creating realistic 3D representations from simple photographic inputs. At the same time, rapid developments in the field of large language models enable natural speech interactions. The overlap of these technologies thus presents an opportunity to create intelligent, visually realistic avatars that are capable of conducting meaningful conversations.

1 State of the Art

3D Gaussian Splatting represents a revolutionary change in the neural scene model. Unlike NeRF, which uses implicit functions, Gaussian Splatting directly represents the scene as a collection of 3D Gaussian elementary graphics elements.

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

Each Gaussian is defined by several parameters: its position in 3D space; a covariance matrix that defines its shape and orientation; a color that is expressed using spherical harmonics for angle-dependent effects; and a transparency value.

Rendering a 3DGS scene involves mapping each 3D Gaussian onto the image plane, where they are subsequently converted into 2D Gaussian points. These 2D Gaussian points are then combined using alpha blending in conjunction with a depth sequence. This process, called splatting, is highly parallelizable and can be efficiently used with modern graphics processors. The data pipeline for creating a 3DGS model begins with capturing a series of photographs of the subject we want to display from multiple angles. These photographs must overlap sufficiently so that the "structure from motion" algorithms can estimate the photographic poses and reconstruct the underlying scattered point cloud [1].

The result is a representation that achieves quality comparable to NeRF while also offering faster rendering. Real-time frame rates are thus achievable even for complex scenes, enabling the use of 3DGS in VR applications [2].

The most commonly used tool for this purpose is COLMAP, which extracts features using SIFT features, assigns individual features across all images, and uses bundle adjustment, a technique that optimizes camera parameters and 3D point positions. The scattered point cloud from COLMAP serves as the starting point for Gaussian Splatting optimization. Each such point becomes the center of a Gaussian, with the underlying covariance and color parameters estimated based on local image statistics. The optimization then refines these parameters and adaptively adds or removes Gaussian points to improve the quality of the image reconstruction [3].

Several tools can be used to simplify the data flow for end users. Polycam is an application that automates the process of capturing photos. It guides users to capture photos from appropriate angles and then automatically creates an image reconstruction [4].

Talking head models are a rapidly developing field that connects several fields such as computer vision, deep learning, and speech synthesis. These models are animated, human heads that speak and express emotions with high visual realism [5].

Lip sync is a field that deals with the production of speech in animated characters so that their lip movements match the phonemes in audio recordings [6].

To overcome the limitations associated with specific languages, auxiliary representations based on phonemes and visemes have been proposed. An example is the framework The Multilingual Experts (MuEx), which uses an architecture based on phoneme-driven expert mixing to ensure strong audiovisual matches [7].

Existing approaches to talking head avatar synthesis can be categorized into five main groups based on their underlying methodology. Each approach differs in terms of visual quality, computational complexity, and suitability for real-time applications, as summarized in the (Tab.1) [8].

Table 1. Overview of methods for synthesizing talking avatars

Type	Description	Advantages	Limitations
2D GAN	Generation via adversarial networks	Visual realism, speed	Training instability, limited to 2D
Diffusion models	Iterative denoising process	Identity preservation, temporal consistency	Slow, expensive
3D morphable models	FLAME, MetaHuman	Geometric consistency, full facial control	Requires 3D data, lower photorealism
NeRF	Neural 3D scene representation	High perceptual quality	Slow rendering, not for real-time
Gaussian Splatting	Explicit 3D Gaussian representation	Real-time rendering, photorealism	No native facial animation support

2 Design of Application

The aim of this paper is to explore the possibilities of creating a VR application implementing a talking avatar created using Gaussian Splatting technology in the Unreal Engine environment. We want to achieve this goal by creating an interactive application that allows the user to communicate with an avatar connected to a large language model. This application is to be integrated into Unreal Engine and into virtual reality using the Meta Quest device.

2.1 Methodology

The system was designed iteratively, rather than comprehensively at the beginning. The application was created by gradually adding components and each iteration of the development had a firmly established and functionally defined goal. Decisions on the selection of individual technologies were made based on the results of previous iterations. This procedure allowed for the acceptance or rejection of individual tools and technologies after practical verification, like detected incompatibility with particular components.

2.2 System Architecture

The development environment consists of several key elements. The entire prototype is developed on macOS Tahoe 26.2 based on Apple Silicon M1 Max. The graphical output of the application is provided by the Unreal Engine 5 game engine. A local Python FastAPI server was used for the backend. The physical availability of the VR headset was limited during development, and therefore the functionality was primarily developed and verified on the desktop in Unreal Engine 5.3.2. VR output is planned as a final validation, not an ongoing part of development.

The designed system consists of two main parts, namely the frontend implemented in Unreal Engine and the backend as a locally running server. Communication between these two entities is carried out using the HTTP protocol.

The frontend provides for the display of the GS avatar, the capture of audio input from the user and the playback of audio output. The backend processes the entire conversational part of the application, namely speech transcription, response generation, speech synthesis and data for lip synchronization. The final visual output of the imported GS avatar can be seen in (Fig.1).

Fig.1. GS avatar rendered in Unreal Engine 5.

The data flow of the conversational application begins when the user presses the K button in the Unreal Engine application. The entire system architecture is shown as a diagram in (Fig.2).

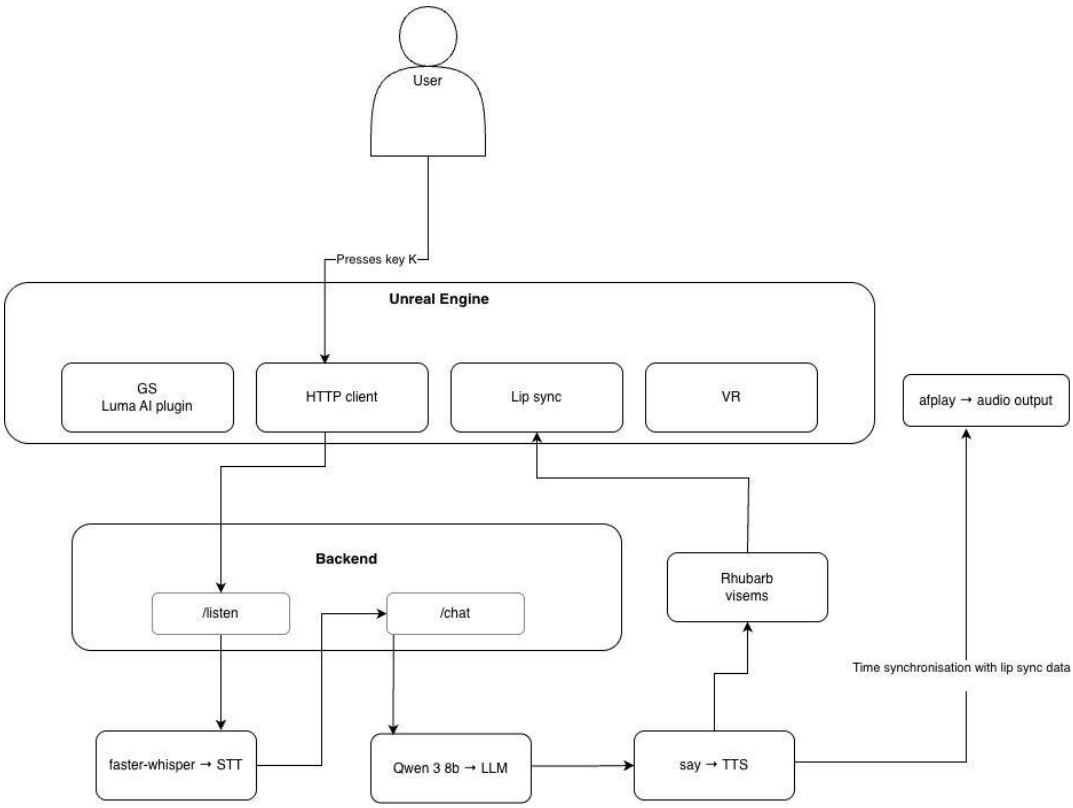

Fig.2. Architecture diagram of the application.

2.3 Testing methodology

System testing was divided into two separate parts. The first part focused on measuring the latency of individual components of our application. Each component was measured separately in order to identify dominant sources of high latency. The overall response was measured by repeatedly executing the data pipeline, with the measurement starting from the moment the user launched the data flow and ending at the moment when both audio and lip sync data were available.

Table 2. Latency measurements

Component	Average	Range	Note
HTTP round-trip	7ms	3 - 15ms	Local, negligible
LLM	2.5s	2 - 3,5s	Depends on the length of text
TTS	0.8s	0,5 - 1,2s	Depends on the length of text
FFmpeg AIFF -> WAV/MP4	0.25s	0.1 - 0.35s	
Rhubarb	0.6s	0,4 - 0,9s	Depends on the length of audio
Total	4.3s	3,2 - 6,2s	

The second part of the testing focused on comparing language models. Since the choice of model directly affects the quality of the user experience, manual comparative testing on a fixed set of prompts was designed. To make the results comparable across models, each model was given an identical system prompt and the same set of test questions. The questions were designed to cover various aspects of the model's behavior relevant to the given use case, in particular the quality of answers in the domain, language support, persona preservation and the model's resistance to prompt injection. The models were tested via the native interfaces of each individual model, either via a web interface or a terminal.

ChatGPT performed the worst in the test, which could be due to the fact that it was a free version, while the Claude models were paid versions. Both Claude models were flawless in the test. The Qwen3-8B model was identical in quality in English questions, but in the case of Slovak questions there was a slight decrease in the quality of the answers. One example is an answer in English to a question in Slovak. The assumption is that a better result could be achieved with appropriate prompt engineering. Since local use was preferred due to the reduction of the overall latency of the application, the local Qwen3-8B model was chosen for the purposes of this work. The summary of the results can be seen in (Tab.3).

Table 3. LLM model comparison.

Model	Quality of answers	Persona preservation	Open-source
Qwen3	5	yes	4
Claude Haiku	5	yes	5
Claude Sonnet	5	yes	5
ChatGPT (free)	4	no	3

3 Discussion

The tool selection for this project was achieved by searching for standard publicly available tools and then testing the functionality locally. For several tools, alternatives had to be chosen, as they were incompatible versions, either in relation to the operating system or a specific version of the Unreal Engine.

Several plugins were tested for rendering the Gaussian Splat avatar in the Unreal Engine. The XV3DGS plugin did not display the UI import after installation in UE 5.5, making it practically unusable. Although UnrealSplat supported UE 5.5, it only contained Windows binaries - this plugin could not be compiled on macOS. The only functional solution was the Luma AI plugin, which officially supports macOS and Unreal Engine 5.3. It was precisely because of this compatibility that the project was moved from UE 5.5 to version 5.3.2. No available GS plugin was able to work on macOS in a newer version of the engine, so a downgrade was necessary.

The backend is implemented as a FastAPI application running on Python 3.11.10. The Python version is managed via pyenv and each project runs in its own virtual environment, which prevents incompatibility between individual libraries.

Local inference via Ollama with the Qwen 3 8B model using a local REST API was chosen as the language model. Compared to the tested models, it showed sufficient quality of responses while allowing local operation without dependence on cloud services.

Piper TTS was originally considered for speech synthesis. However, it was rejected because its repository was archived and its macOS compatibility was broken. The native say tool, which is part of macOS, was used as a replacement. The voice quality is lower than with dedicated TTS tools, but the solution does not require any external dependencies. This compromise was acceptable for the purposes of the paper.

The measured latency is on the edge of acceptability for an interactive VR experience. Improvements would be achieved by replacing the local model with a larger or faster model, more powerful HW, or optimizing Rhubarb generation with parallel processing.

The most serious problem of the resulting solution is related to the limitation of GS representation for animation in Unreal Engine through available plugins. These do not allow graphical modification of the inserted model, but consider the given model as a whole. Since some of the available plugins are open-source projects, one solution could be to modify the code. This modification would have to ensure that relevant parameters are exposed and modifiable in Unreal Engine, which would allow graphical manipulation. However, this step is beyond the scope of this thesis.

Another approach to animating avatars is described in the GaussianAvatars method, where Gaussians are bound to a FLAME mesh and trained together from scratch on a video taken from multiple angles [9]. However, this method is outside of the scope for our paper, as the implementation was carried out exclusively in Unreal Engine without access to similar training infrastructure.

Conclusion

The aim of this study was to investigate whether it is possible to create a functional real-time conversational avatar using Gaussian Splatting in the Unreal Engine environment. The result of the work is a functional prototype that verifies this question in practice.

A system consisting of two main parts was designed and implemented, namely a backend built on FastAPI, as well as a client application in Unreal Engine 5. The backend provides the entire conversational data stream: speech recognition via the faster-whisper tool, response generation using the local language model Qwen 3 8B via Ollama, speech synthesis, audio format conversion and lip sync data generation using the Rhubarb tool. The Gaussian Splatting avatar is rendered in real-time using the Luma AI plugin, while lip synchronization could not be solved. The system was successfully integrated into the VR environment using the UE plugin OpenXR.

The work verifies the technical feasibility of combining a Gaussian Splatting avatar and a real-time conversational LLM data stream. This is an area for which there were no established and standardized solutions at the time of this work.

The resulting prototype can serve as a basis for future interactive installations combining photorealistic representation of characters with conversational AI systems. A positive contribution of this work is also the test of large language models, the more detailed results of which can be found in Appendix 1. These results can also be used outside the domain of interactive conversational applications.

The most significant limitation remains the inability to easily implement lip animations. The overall pipeline latency, averaging around 4.3 seconds, is on the limit of acceptability for a smooth conversational experience. The system was developed and tested primarily on a desktop display, with VR testing limited by the availability of a headset. Due to incompatibility, audio playback was handled on the local server side, which represents an architectural limitation when deployed on another platform, and for future development it would be more appropriate to implement audio playback directly in Unreal Engine.

The immediate next step is to complete the lip sync integration. A feasible solution is to implement sprite overlay visemes directly in the Unreal Engine Blueprint logic based on data generated by the Rhubarb tool. Consequently, the greatest potential for improvement is to replace sprite overlay with a more sophisticated method. This could be, for example, an extension or modification of available open-source plugins that would allow Gaussian Splatting avatar deformation and animation directly in Unreal Engine, or it could be the use of one of the currently developed or researched solutions for animated GS avatars.

Acknowledgement

The author would like to thank RNDr. Ján Lacko, PhD. for his support during the development of this paper. Additionally, the author wishes to express gratitude to the Faculty of Informatics at Pan-European University for providing the resources and academic environment to complete the study. This paper was conducted as part of the diploma thesis titled AI-based deep fake avatars using Gaussian Splatting.

References

[1] Bernhard Kerbl, Georgios Kopanas, T. Leimkühler, and G. Drettakis, (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–1, doi: https://doi.org/10.1145/3592433.

[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. arXiv:2003.08934.

[3] J. Schönberger and J.-M. Frahm, (2016). Structure-from-Motion Revisited. Available: https://demuc.de/papers/schoenberger2016sfm.pdf.

[4] Polycam, (2024). Polycam - LiDAR 3D Scanner. Available: https://poly.cam/.

[5] H. Nisar, S. Masood, Z. Malik, and A. Abid, (2026). Talking Head Generation Through Generative Models and Cross-Modal Synthesis Techniques, *Journal of Imaging*, vol. 12, no. 3, p. 119, doi: https://doi.org/10.3390/jimaging12030119.

[6] N. H. Loh, (2014). Development of Real-Time Lip Sync Animation Framework Based On Viseme Human Speech, *Archives of Design Research*, vol. 112, no. 4, p. 19, doi: https://doi.org/10.15187/adr.2014.11.112.4.19.

[7] Caglar, E., Oskooei, A. R., Kutanoglu, M., Keles, M., & Aktas, M. S. (2025). Asynchronous Pipeline Parallelism for Real-Time Multilingual Lip Synchronization in Video Communication Systems. arXiv preprint arXiv:2512.18318.

[8] H. Nisar, S. Masood, Z. Malik, and A. Abid, (2026). Talking Head Generation Through Generative Models and Cross-Modal Synthesis Techniques, *Journal of Imaging*, vol. 12, no. 3, p. 119., doi: https://doi.org/10.3390/jimaging12030119.

[9] Qian, S., Kirschstein, T., Schoneveld, L., Davoli, D., Giebenhain, S., & Nießner, M. (2024). Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 20299-20309).

Author

Bc. Ondrej Hudcovič
Faculty of Informatics
Pan-European University in Bratislava, Slovakia
xhudcovic@paneurouni.com
Data Platform Engineer with background in Information Systems and AI. Currently studying Applied Informatics at the Faculty of Informatics at Pan-European University in Bratislava.

International Journal
Information Technology Applications (ITA)

Volume 15, Number 1, July 2026

HOW THE PRESENCE OF GAME PATTERNS AFFECTS GAMES

Ivana Milovanović

Abstract:

Game patterns represent fundamental components of video game design that influence player behavior, engagement, progression, and long-term retention. While these mechanics are often implemented to improve player satisfaction and motivation, certain implementations may evolve into dark patterns that create psychological pressure or encourage excessive engagement. This article examines selected game patterns commonly found in modern live-service games and analyzes their influence on player experience. Through a comparative analysis of two games with similar progression and monetization structures, the study explores how different implementations of mechanics such as stamina systems, character presentation, animations, and exploration mechanics affect player satisfaction and behavior. The findings demonstrate that the design and implementation of game patterns significantly impact player perception, highlighting the importance of balancing engagement-focused mechanics with player-friendly design principles.

Keywords:

Game patterns, dark patterns, game design, player experience.

Introduction

Although they are not commonly being explicitly mentioned as such, game patterns represent constitutive elements of video games. For example, Playing by Appointment is a game pattern that implements mechanics designed to encourage players to make habitual interactions with the game [1, 15].

Game patterns affect how players interact with the game, how often they return, and how emotionally connected they feel to the experience. These patterns can improve player engagement, motivation, and satisfaction, but they can also create frustration, pressure, or unfair advantages when implemented poorly.

This article explores how the presence of specific game patterns affects two similar games. The games in question are Genshin Impact [4] and Honkai: Star Rail [5], developed by the same company, HoYoverse. Even though both games share the same progression structures and monetization systems, different game patterns have different player reactions and experience.

The goal of this article is to analyze both positive and negative effects of selected game patterns, examine how they influence player behavior, and compare how each game approaches these mechanics. These game patterns were selected because they are commonly used in modern live service games and because they affect player satisfaction.

The rest of this article is organized as follows. Section 1 reflects on the position of game patterns. Each of Sections 2–6 is dedicated to one game pattern and its effects on the two studied games. The last section concludes the article.

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

1 Game Patterns

Game patterns are recurring gameplay mechanics and systems used by developers to shape player behavior and structure the gameplay experience. They represent fundamental elements of video game design and appear across many different genres and game types, including educational games [11]. These patterns influence how players interact with games, how progression is structured, and how long players remain engaged with the experience.

Developers use game patterns to improve immersion [12], player retention, motivation, and satisfaction. Mechanics such as stamina systems, limited-time events, reward structures, and progression systems are commonly implemented to encourage regular player activity and long-term engagement.

However, not all game patterns affect players positively. Some game patterns can become dark patterns. Dark patterns are design solutions intentionally created to manipulate player behavior in ways that primarily benefit the developer rather than the player. These mechanics often rely on psychological pressure, fear of missing out, repetitive reward loops, or artificial limitations designed to increase player retention, spending, or daily activity.

In order to explore them, game patterns have to clearly expressed. The problem expressed as a conflict of the two most prominent (or main) contradicting forces is essential for this [8, 13, 14], so this approach is used in this article, too.

2 Playing by Appointment

Appointment mechanics are a great way to make players want to keep coming back to a game, but players get frustrated on missing free stamina that is needed to progress the game.

Forces: Appointment mechanics are a way to make players want to keep coming back to the game, but players get frustrated on missing free stamina that is needed to progress the game.

Resolution: Implement time-based stamina systems that gradually regenerate over time and allowing stamina overflow or long-term storage systems.

First mentioned by Zagal et al. [15], Playing by Appointment is a dark pattern that requires that players play at specific times (or dates) as defined by the game rather than by the players. An example of this would be stamina systems in games. This system limits a person's playtime, as they are required to expend some sort of resource, in this case, stamina in order to do content in the game. Players use up their stamina and generally are not allowed to do any meaningful content afterwards. As time passes their stamina refills, and they can keep playing and progressing the next day. This by itself is not necessarily a dark pattern, because it is common practice for certain games, especially massively multiplayer online (MMO) games, to limit player playtime in one way or another.

However, this can quickly become a dark pattern [2] if the design of the stamina system itself is predatory. For example, a stamina system that respects a player's time would allow for them to miss a few days of playing and still progress the same amount when they do get to play. Their stamina would fill back up and then overcap, based on how much time they have not played. In the case of the two games that we are exploring, Genshin Impact doesn't automatically overcap. It requires the player to open the game and convert the short term stamina ("resin," as it is called in the game) into long term stamina ("condensed resin"). Compared to the other game, Honkai: Star Rail, capping is done automatically, and the player doesn't feel the pressure to open the game. The time difference before the player is not punished for not playing is also different.

In the Genshin Impact case it is twenty-six hours (see Fig.1) if a player doesn't open the game and roughly two and half days if they do. In the Honkai: Star Rails case it goes up to thirty hours and thirty days (see Fig.2).

Fig.1. Genshin's resin system, fully fills after twenty-six hours of not playing.

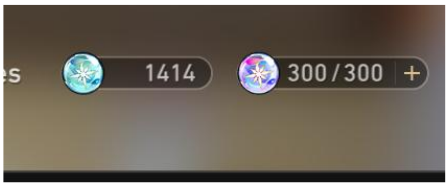

Fig.2. Honkai: Star Rails trailblazer power system, up to thirty days and thirty hours.

3 Limited Time Events as a Reason to Come Back

Limited time events are a staple in live service games. They are the easiest way to keep the game fresh and content entertaining, but some players can't participate in every event that they want to, which may trigger the fear of missing out [6].

Forces: Limited-time events provide fresh content and rewards, encouraging players to return regularly to the game, but players who miss events may feel punished or excluded from unique rewards and story content.

Resolution: Create recurring limited time events while allowing players to access missed content later through permanent event archives or alternative reward systems.

In Genshin Impact, there are usually four events per patch, three small events and one bigger one. The problem with that is that players who don't log in the time frame that event is running, they might miss out on a limited time reward that can't ever be unlocked again. Players who unlock it feel more special, but the rest of players feel the pressure and frustration that they are missing out on content. In Honkai: Star Rail, events are kept forever in a special tab of the game, so players who miss out events are not forever locked out of the rewards that were in it. They also reward players who complete it with extra rewards, usually currency, which can easily be earned in other parts of the game.

The impact of limited time events and exclusive rewards has been studied by Lorriane [6], but without concisely expressing the findings as a game pattern.

4 Appearance of Character Abilities Follows Its Tier

It's only natural that players are more attracted by better looking characters.

Forces: Clear visual and gameplay differences between lower tier and high tier characters create strong player motivation to obtain higher rarity units, but players may feel that lower tier characters are less valuable, reducing long-term engagement with lower rarity units.

Resolution: Give all characters visually satisfying abilities and animations while still preserving distinct visual quality and prestige for high tier characters.

What distinguishes the characters in different tiers is how easy it is to get them [3]. For the players who want to get a lower tier character, they can get it easily in around a few hours of game time. Sometimes, the game also awards players with those lower tier characters during the events. On the other hand, to get a high tier character, players generally have to spend two to four weeks of active play to get a high tier character. This is true for both games that we are exploring in this article.

The difference in quality of those characters is that in Genshin Impact, lower tier characters don't feel as significant and flashy (see Fig. 3) as the high tier characters are (see Fig. 4). High tier characters get animations for their ultimate ability, while lower tier characters don't. In Honkai: Star Rail, both tiers get flashy animations that make all characters feel special (see Fig.5 and 6).

Fig.3. Low tier character (Layla) in Genshin Impact doing the ultimate ability animation.

Fig.4. High tier character (Arlecchino) in Genshin Impact doing the ultimate ability animation.

Fig.5. High tier character (Gallagher) in Honkai: Star Rail doing the ultimate ability animation.

Fig.6. High tier character (Sparkle) in Honkai: Star Rail doing the ultimate ability animation.

5 Animation Design

Animations are one of the main selling points for high tier characters, making their visual presentation extremely important for player satisfaction. Developers aim to create animations that feel impactful, cinematic, and visually rewarding during combat while also ensuring that they remain enjoyable after repeated use. Designing animations that are both exciting and efficient is important, since players may watch the same abilities hundreds of times throughout regular gameplay.

Forces: High quality animations and cinematic ultimates enhance satisfaction and emotional payoff during combat, but overly long or flashy animations can slow gameplay pacing and become repetitive over time.

Resolution: Implement cinematic combat animations while providing options such as animation skipping or combat speed adjustments to maintain gameplay flow.

Powerful looking animations will bring player satisfaction during combat. Flashy effects, cinematic camera angles, and impactful ultimate abilities create satisfying moments during combat and make characters feel stronger and more rewarding to obtain.

However, because these animations are triggered repeatedly during gameplay, some players may eventually find them repetitive or disruptive to fast paced combat. Having alternative shorter animations or having the ability to completely skip it during the combat. Both games have this problem, but in Honkai: Star Rail you have the ability to speed up the combat, which speeds up the animations.

6 Exploration Movement Restrictions

Exploration is one of the most important aspects of open world games, which makes movement systems an essential part of the player experience. Developers aim to create traversal mechanics that feel immersive and realistic by connecting sprinting, climbing, swimming, and gliding to a stamina system. At the same time, movement should remain smooth and enjoyable over long play sessions, without making players feel unnecessarily restricted or punished while exploring the world.

Forces: Movement stamina systems make exploration feel immersive, realistic, and strategically engaging, but constant stamina limitations can interrupt exploration flow and make traversal feel frustrating or artificially restrictive.

Resolution: Balance realistic traversal mechanics with movement freedom by reducing stamina frustration or minimizing the punishment of mismanaging the stamina.

In the open world games, such as Genshin Impact, traversal throughout the world is important to get a good feel of. Transition between sprinting, climbing, swimming and gliding with a stamina bar feels realistic and also encourages players to think strategically about traversal routes. However, movement limitations can also negatively affect the pacing of exploration. Constantly stopping to recover stamina may interrupt immersion and make traversal feel unnecessarily slow, especially during long exploration sessions or mountain climbing. Genshin Impact relies on exploration as a core mechanic in the game, but the stamina bar feels frustrating to play with, while exploration in Honkai: Star Rail feels fast and efficient without a stamina bar, it feels too short and players are not satisfied with it.

The Bird Cage pattern [7] also targets gliding. However, the pattern presented here is more general.

Conclusions

This article analyzed both positive and negative effects of different game patterns that influence player retention and satisfaction. Understanding these patterns is important because modern live service games rely heavily on them to maintain long term player activity and monetization.

The article also highlighted how some patterns may evolve into dark patterns when they create unnecessary pressure, fear of missing out, or artificial gameplay restrictions.

Overall, this article shows that game patterns are not inherently positive or negative. Their effect depends largely on how they are implemented and balanced. Well balanced patterns can improve immersion, motivation, and player enjoyment, while poorly balanced implementations may create frustration and reduce long term satisfaction.

It may be worthwhile to explore the effects of the fundamental properties as stated by Alexander [1], similarly as they have been used to assess user interfaces [9, 10].

Acknowledgments

I would like to express my sincere gratitude to professor Valentino Vranić from Pan-European University in Bratislava, Slovakia. Special thanks belong to Sinergija University in Bijeljina, Republic of Srpska, Bosnia and Herzegovina for supporting my studies.

References

[1] Christopher Alexander. The Nature of Order: An Essay on the Art of Building and the Nature of the Universe, Book 1 – The Phenomenon of Life. The Center for Environmental Structure Publishing, Berkeley, CA, USA, 2002.

[2] Lena Emara. Playing the Game of Dark Patterns. Medium, 2020. https://medium.com/@lena.emara/playing-the-game-of-dark-patterns-d20c7dd2ff04

[3] Heathrileyo. Rarity in Game Design — Why Some Cards (and Characters) Just Feel Special. Medium, 2025. https://medium.com/@Heathrileyo/rarity-in-game-design-why-some-cards-and-characters-just-feel-special-d1739a8ab232

[4] HoYoverse. Genshin Impact, 2026. https://genshin.hoyoverse.com/en/game

[5] HoYoverse. Honkai: Star Rail, 2026. https://hsr.hoyoverse.com/en-us/home

[6] Shophia Lorriane. The Impact of Limited-Time Events and Exclusive Rewards. EasyChair Preprint 13006, 2024. https://easychair.org/publications/preprint/Cr9Xc

[7] Tanuj Rao. The Bird Cage. Pattern Language for Game Design: Pattern Library. https://patternlanguageforgamedesign.com/PatternLibraryApp/PatternLibrary/4068

[8] Aleksandra Vranić and Valentino Vranić. Understanding Design Patterns Through Drama Metaphors. In Proceedings of the 2024 IEEE 17th International Scientific Conference on Informatics, Informatics 2024. Poprad, Slovakia, IEEE, 2025.

[9] Branislava Vranić. Assessing User Interface Design by Alexander's Approach to Building Architecture. In Proceedings of the 32nd Conference on Pattern Languages of Programs, People, and Practices, PLoP 2025. Skamania Lodge, Columbia River Gorge, WA, USA. ACM, 2026.

[10] Branislava Vranić. Creating User Interfaces with Alexander's Fundamental Properties. In Proceedings of the 32nd Conference on Pattern Languages of Programs, People, and Practices, PLoP 2025. Skamania Lodge, Columbia River Gorge, WA, USA. ACM, 2026.

[11] Branislava Vranić and Júda Vodrážka. Mixing Educational Value and Entertainment Games Through Patterns. In Proceedings of the 32nd Conference on Pattern Languages of Programs, People, and Practices, PLoP 2025. Skamania Lodge, Columbia River Gorge, WA, USA. ACM, 2026.

[12] Branislava Vranić and Valentino Vranić. Patterns of Recreating Reality in Games. In Proceedings of the 29th Conference on Pattern Languages of Programs Online, PLoP 2022. ACM, 2023.

[13] Valentino Vranić. Drama and Software Patterns. In Proc. of the 2024 IEEE 17th International Scientific Conference on Informatics, Informatics 2024. Poprad, Slovakia, IEEE, 2025.

[14] Valentino Vranić and Aleksandra Vranić. Drama Patterns: Extracting and Reusing the Essence of Drama. In Proceedings of the 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019. Kloster Irsee in Bavaria, Germany. ACM, 2019.

[15] José Pablo Zagal and Staffan Björk, and Chris Lewis. Dark Patterns in the Design of Games. In Proceedings of the 8th International Conference on Foundations of Digital Games, FDG 2013. Chania, Crete, Greece, 2013.

Author

Ivana Milovanović
Faculty of Computing and Informatics
University Sinergija in Bijeljina, Republic of Srpska, Bosnia and Herzegovina
Ivanask467@gmail.com

Ivana Milovanović is interested in programming, data analysis, and web technologies, with a focus on developing technical and analytical skills. She explores software development, working with data, and creating modern web solutions.

International Journal
Information Technology Applications (ITA)

Volume 15, Number 1, July 2026

PROPOSAL AND IMPLEMENTATION OF CONVERSATION AI SYSTEM FOR AVATARS IN VIRTUAL REALITY

Roman Zemaník

Abstract:

This article presents the design and implementation of a conversational AI system for avatars in virtual reality. The system enables spoken interaction with MetaHuman avatars in Slovak and English and is implemented in Unreal Engine 5. The work connects speech recognition, large language model response generation, speech synthesis, audio playback, avatar state feedback, and lip-sync animation. Existing local and cloud-based models are integrated into a modular STT-LLM-TTS pipeline so that different processing configurations can be compared. The evaluation focuses on response latency, hardware load, and the selection of a suitable configuration for VR use. Technical testing showed that the lowest average latency was achieved by the configuration using cloud STT, local LLM, and local TTS with an average response time of 1.94 s. For user testing, cloud STT, cloud LLM, and local TTS were selected as a better compromise between response quality and latency. The results show that LLM-based conversational AI can be integrated with VR avatars in Unreal Engine while preserving flexibility for further model and performance evaluation.

Keywords:

Virtual reality, conversational avatar, speech recognition, text-to-speech, large language models, Unreal Engine.

Introduction

Virtual reality allows the user to immerse themselves in a digital environment and perceive it as a space around them. Therefore, forms of interaction that resemble communication in the real world feel more natural in VR. Traditional text input through a virtual keyboard or selection from predefined dialogue options can interrupt the intended experience. One way to bring VR interaction closer to natural communication is through conversational avatars capable of responding to the user's free spoken input.

The implementation of such conversational avatars can be based on large language models, which can process freely formulated user input and generate responses in natural language. However, the language model alone is not sufficient for creating spoken interaction in virtual reality. It must be connected with speech recognition, speech synthesis, and the visual representation of the character so that the user can conduct a conversation directly in the virtual environment. Important factors are therefore not only the correctness of the answers, but also the response speed of the system and the visual presentation.

The aim of this work is to design and implement a prototype of a conversational avatar in virtual reality. The system is built in Unreal Engine 5, uses MetaHuman to create a realistic avatar, and supports voice communication in Slovak and English. The conversational pipeline is divided into separate parts for speech recognition, text response generation, and speech synthesis. The implementation also allows local and cloud-based solutions to be combined so that different combinations of these components can be tested.

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

The work first focuses on an overview of existing approaches to creating conversational avatars in virtual reality. Then the system design, its requirements, architecture, and the selection of used technologies are described. The implementation chapter describes how the system captures the user's voice input, processes it in the conversational pipeline, and presents the response through the avatar's voice and visual behavior. The final chapters focus on technical and user testing of the implemented system and possibilities for further extension.

1 Overview

Conversational agents developed from rule-based text systems to embodied agents in virtual reality. ELIZA used keyword matching and predefined response patterns without semantic understanding [1]. Later systems introduced more advanced rule-based methods and early natural language processing. STEVE was one of the early embodied pedagogical agents used in a virtual environment [2]. Current conversational-agent research describes a shift toward generative systems based on large language models (LLM), which can produce responses from conversation context [3].

Simulating human interaction with avatars requires several connected mechanisms. The system must interpret user input, generate responses, process speech in real time through STT (Speech-To-Text), LLM, and TTS (Text-To-Speech), and preserve at least short-term conversational context [4], [5]. Embodiment, personality modeling, facial animation, lip-sync, and state feedback influence how natural and readable the avatar feels [6]. Spatial context is also important, because questions in VR can depend on objects, gaze, pointing, and timing [7].

Applications with LLM avatars usually combine a VR engine, a language model, and a speech pipeline. The engine provides the virtual environment, avatar representation, and interaction logic. The language model generates the avatar's answer, while the speech pipeline converts the user's voice to text and the generated answer back to speech. In several related systems, the speech output is also connected with lip-sync or facial animation so that the answer appears to come from the virtual character [4], [8], [9].

LLM avatars can be used in several types of VR applications. In games, they can extend scripted NPC dialogue and make virtual characters react to the current scene [7]. In education and professional training, they can guide the user through tasks or act as simulated conversation partners [9]. Other examples include productivity assistants that use workspace context, social VR agents that maintain longer conversations, and virtual characters used in public events or self-talk scenarios. These use cases differ in purpose, but they rely on the same basic idea: spoken interaction with an embodied character.

Although LLM avatars enable more natural and flexible interaction than traditional systems, their use in VR introduces several limitations. Slow responses reduce the fluency of conversation and can frustrate users [10]. Cloud services can improve quality and simplify integration, but they introduce internet dependence, API costs, and external processing. Local models reduce these dependencies, but they increase hardware requirements. Other limitations include restricted context and memory, hallucinated or factually incorrect answers, weaker control over the avatar's role-specific behavior, and the need to connect LLM, STT, TTS, avatar animation, and VR interaction manually [4], [5].

The quality of an LLM avatar in VR cannot be evaluated only by whether the model generates linguistically correct answers. In related work, evaluation is usually divided into technical parameters of the system and the user experience of interaction. Technical evaluation focuses mainly on response time, stability, consistency, and the quality of generated answers. User-oriented evaluation focuses on naturalness of interaction, social presence, immersion and overall user satisfaction [6], [10].

2 Design of the Conversational Avatar

The system requirements follow the goal of voice-based interaction with an avatar in virtual reality. The user should communicate through a microphone without text input or predefined dialogue options. The system must support Slovak and English, produce answers in the corresponding language, keep response latency low enough for interactive use, and measure the duration of individual processing steps. The generated answer should be presented through voice, avatar visual feedback, and lip-sync, not only as text.

One interaction starts when the user asks the avatar a question by voice (Fig.1). The application records the input, sends it to the conversational system, and the avatar switches from idle to thinking while the request is processed. The conversational system contains three separated modules: STT converts speech to text, LLM generates the answer, and TTS converts the answer back to speech. After synthesis, the avatar plays the generated audio, switches to the talking state, applies lip-sync, and returns to idle after playback. This architecture separates conversational logic from avatar presentation and allows local and cloud STT, LLM, and TTS components to be compared without changing the interaction flow.

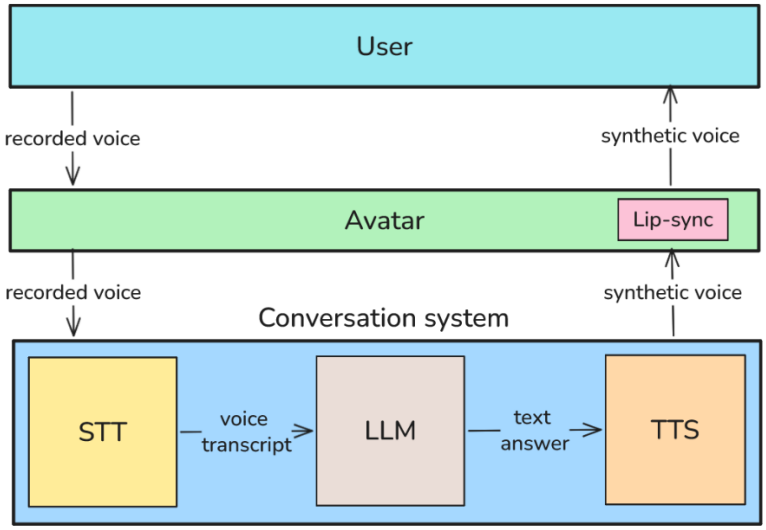

Fig.1. Conversation system.

After defining the expected behavior and architecture of the system, it was necessary to select the technologies used for implementation. The selection considered compatibility, latency, output quality, ease of integration, and practical usability for creating a functional prototype. Unreal Engine 5.7 was selected because it supports OpenXR for VR development and provides a prepared VR template. MetaHuman was selected for the visual representation of the avatar because it allows the creation of a realistic human character without modelling it from the beginning. Meta Quest 3 was selected as the VR headset because it provides a built-in microphone, stereoscopic display, headset and controller tracking, hand tracking, and compatibility with Unreal Engine 5 through OpenXR.

For the conversational pipeline, the selected cloud models were gpt-4o-mini-transcribe for speech recognition, gpt-4.1-nano for response generation, and gpt-4o-mini-tts for speech synthesis. The selected local speech recognition model was faster-whisper-small, because it provided the best compromise between latency and transcription accuracy.

Local speech synthesis was implemented with Piper using sk_SK-lili-medium for Slovak and en_US-norman-medium for English. For local response generation, qwen3-4b-2507 Q8_0 was used because it was compatible with the selected Unreal Engine plugin.

3 Implementation of the System

The first implementation step was creating the Unreal Engine project from the VR template. The template provided the basic VR player pawn, teleport movement, controller input, and settings needed to run the application in VR mode. This made it possible to focus on the conversational part of the system and avatar integration instead of implementing basic VR controls from the beginning. The project was developed and tested in the Unreal Engine editor using VR Preview. External plugins were added for cloud model integration, local language model inference, and lip-sync processing.

MetaHuman Creator was used to prepare the avatars that represent the conversational characters in the VR scene. Instead of creating characters manually from the beginning, prepared MetaHuman presets were used and adjusted for the needs of the prototype. The avatars were exported in the UE Optimized quality setting, because the system is intended for an interactive VR application where performance is important. After the characters were created, animation assets for the avatar states were prepared and retargeted to the MetaHuman skeleton. The avatars were then placed into the virtual scene, creating the visual basis for the conversational system.

Several Unreal Engine plugins and modules were added to support the conversational pipeline. The UnrealOpenAIPlugin [11] plugin was used for communication with cloud STT, LLM, and TTS models, Llama-Unreal [12] was used for local language model inference, and OVRLipSync [13] was used to generate visemes for lip-sync animation. Cloud processing was prepared through the OpenAI Platform, where an API key was created for access to the selected cloud model. Local speech recognition and local speech synthesis were implemented in a separate FastAPI Python server instead of directly inside Unreal Engine. The Unreal Engine application communicates with this server through HTTP requests.

The main implemented components are (Fig.2) BP_XRPawn, ConversationalAvatarComponent, ConversationSubsystem, STTService, LLMService, and TTSService. BP_XRPawn records the user's voice and creates an audio file. ConversationalAvatarComponent receives the file from the active avatar and forwards it to ConversationSubsystem. ConversationSubsystem then calls STTService, LLMService, and TTSService in sequence. The generated WAV response is returned to ConversationalAvatarComponent, which plays it through the avatar audio component and uses it for lip-sync and animation state changes.

Voice input is captured in BP_XRPawn, which represents the player in the VR scene. An Audio Capture component and recording submix are used to record the microphone signal. Recording can be started manually with the A button on the Quest 3 controller or automatically in Open Mic mode, where recording starts after the microphone volume exceeds a threshold and stops after a period of silence. After recording ends, BP_XRPawn exports the input as a WAV file and passes its path to the currently active ConversationalAvatarComponent. This component is attached to each conversational MetaHuman avatar and stores avatar-specific settings such as active state, name, language, gender, and local or cloud processing options. Before forwarding the recording, the component checks that the WAV file has already been fully written to disk. It then switches the avatar to the thinking state and sends the recording to ConversationSubsystem through GenerateAudioResponseFromAudio().

The conversation flow is controlled by ConversationSubsystem, implemented as a GameInstanceSubsystem available during the whole runtime of the application. During initialization, it creates STTService, LLMService, and TTSService and later coordinates their execution instead of performing the processing directly.

Before each request, it prepares the active configuration by combining global settings from ConversationSettings with avatar-specific values such as language, name, gender, and selected local or cloud processing options. The configuration is divided into STT, LLM, and TTS settings, so each service receives only the values needed for its part of the pipeline. Processing starts in GenerateAudioResponseFromAudio(), where the subsystem receives the recorded WAV file and the active avatar, prepares the configuration, and sends the recording to STTService as the first step of the pipeline.

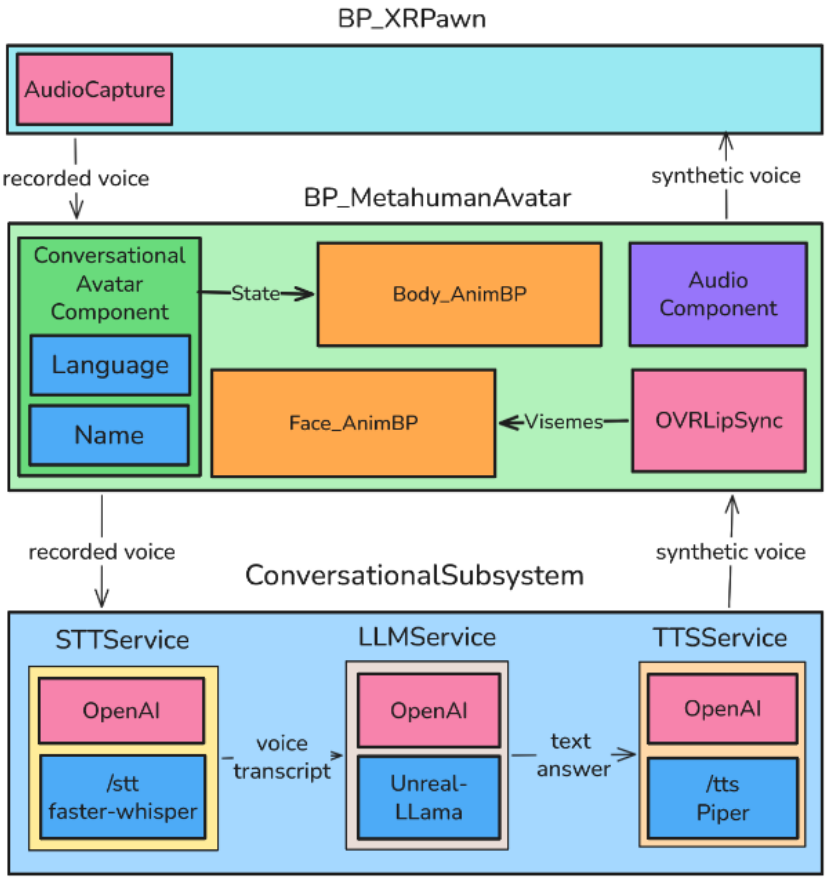

Fig.2. The main implemented components.

The three processing steps are implemented in STTService, LLMService, and TTSService. STTService receives the recorded WAV file, checks that it is available, and converts the user's speech to text using either the local FastAPI server with faster-whisper-small or the cloud model gpt-4o-mini-transcribe through UnrealOpenAIPlugin. The transcript is then passed to LLMService, which generates the avatar's text response either locally through Llama-Unreal with a Qwen model or in the cloud using gpt-4.1-nano. The active system prompt and avatar-specific values such as name, gender, and language are used during generation. The generated text is then passed to TTSService, which creates the spoken response either locally through the FastAPI server with Piper or in the cloud using gpt-4o-mini-tts. The result is saved as a WAV file and returned to ConversationSubsystem, which sends it back to the avatar component for playback.

After the WAV response is generated, ConversationalAvatarComponent presents it in the VR scene. The HandleGeneratedAudioResponse() method ends the thinking state, plays the WAV file through the avatar audio component, and switches the avatar to the talking state during playback. After the audio finishes, the avatar returns to idle. The same WAV file is also used to prepare lip-sync data. Body animation is controlled by Body_AnimBP, which switches between idle, thinking, and talking states and turns the active avatar's head toward the user (Fig.3). Face animation is controlled by Face_AnimBP. The audio is split into 10 ms segments and processed by OVRLipSync, which produces viseme values for each moment of speech. Because MetaHuman does not use OVR visemes directly, the values are mapped to MetaHuman face curves such as jaw opening and lip movement. These curves are applied during playback, while Face_AnimBP also adds simple randomized blinking so that the avatar does not appear static.

Fig.3. Avatar.

A simple VR menu was extended from the existing template menu to support avatar selection and voice input mode switching. The user can select the active avatar, which updates bIsActiveAvatar on the available ConversationalAvatarComponent instances and changes CurrentTargetAvatar in BP_XRPawn. The menu also allows switching between manual recording and Open Mic mode by changing bOpenMic and reinitializing voice recording. ConversationSubsystem also logs each request into a TSV file for later evaluation. The log contains the selected configuration, avatar, processing times for individual pipeline steps, total duration, transcript, generated response, status, and possible errors, which makes it possible to compare local and cloud processing without manual timing.

4 Evaluation

The technical evaluation focused on comparing the latency of different STT, LLM, and TTS combinations and selecting a configuration suitable for user testing. Testing was performed on a computer with an AMD Ryzen 7 7700X CPU, 32 GB RAM, AMD Radeon RX 9070 XT GPU with 16 GB VRAM and Windows 11 Pro. Hardware load during VR runtime was measured with MSI Afterburner in three scenarios: the base VR scene without the conversational pipeline, a cloud-based pipeline, and a fully local pipeline. The base scene and cloud pipeline kept stable FPS, while the fully local pipeline caused visible FPS drops when local models started processing the input. GPU load remained high in all scenarios, which indicates that the VR scene itself was already demanding, while CPU load showed clearer differences because the local pipeline produced short peaks correlated with FPS drops.

The latency evaluation (Fig.4) compared all eight combinations of local and cloud processing for the three pipeline components: STT, LLM, and TTS. The notation uses C for cloud and L for local processing, in the order STT-LLM-TTS. For each request, the system recorded STT time, LLM time, TTS time, and total pipeline time from sending the recorded audio to receiving the synthesized response. The lowest average latency was achieved by C-L-L with a total time of 1.94 s. A similar result was achieved by C-C-L with 2.12 s, while the highest latency was measured for L-L-C with 5.80 s. The results show that cloud TTS increased the total processing time the most, while the difference between local and cloud LLM processing was smaller than the differences caused by STT and TTS choices.

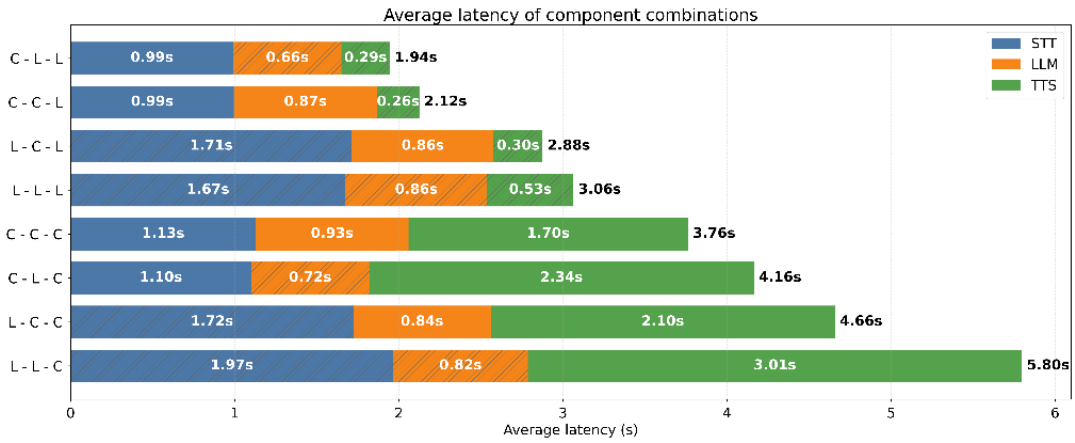

Fig.4. Average latency.

User testing was conducted after technical testing and after selecting the final system configuration. The selected configuration was C-C-L, which used cloud STT, a cloud language model, and local TTS. Each participant first received a short explanation of the test and basic VR controls. During the test scenario, the participant had to ask the avatar at least 10 questions, including at least one follow-up question related to a previous answer. This was used to test both separate responses and the ability to maintain context in a short conversation. After the interaction, each participant filled in a questionnaire based on a 5-point Likert scale, where 1 meant strongly "disagree" and 5 meant "strongly agree". The questionnaire contained 13 questions focused on ease of communication, quality and naturalness of answers, voice output, fluency of interaction, perceived latency, and the overall impression of the conversation in VR.

User testing showed a generally positive response to the system. Participants rated (Fig.5) the interaction as engaging, considered the response time acceptable, and indicated that a similar form of communication could be useful in another VR application. Lower ratings were connected mainly with the naturalness of the answers, the voice output, and lip-sync. Since users were mostly satisfied with the speed of the system but less satisfied with the voice presentation, one possible improvement is to replace the local TTS model with a higher-quality cloud TTS model. This would probably increase latency, but it could improve the naturalness and persuasiveness of the avatar's spoken output.

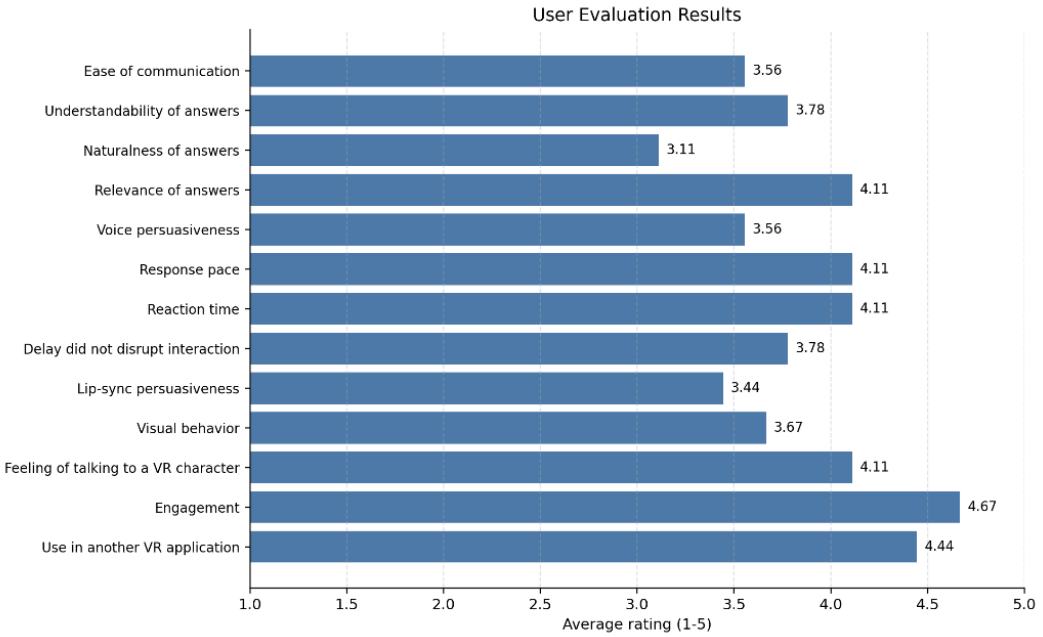

Fig.5. Average rating.

Conclusion

The aim of this work was to design and implement a prototype of a conversational avatar in virtual reality that supports spoken communication in Slovak and English. The system was implemented in Unreal Engine 5 with MetaHuman avatars and a modular conversational pipeline for speech recognition, response generation, and speech synthesis.

The implementation separates the VR player, avatar component, conversation subsystem, and individual STT, LLM, and TTS services. The system supports both local and cloud-based processing, which makes it possible to compare different configurations. Local STT and TTS are provided through a separate FastAPI server, while cloud processing is handled through OpenAI models. The avatar also provides visual feedback through idle, thinking, and talking states, head direction toward the user, blinking, and lip-sync animation.

Technical testing showed clear differences between local and cloud configurations. The lowest average latency was achieved by the configuration using cloud STT, local LLM, and local TTS, but this setup did not provide sufficient response quality in Slovak. Fully local processing also caused higher hardware load during VR use. Therefore, the configuration with cloud STT, cloud LLM, and local TTS was selected for user testing as a better compromise between response quality and latency. User testing showed that participants generally accepted the interaction positively, especially its

engagement and response speed, while the voice output quality and lip-sync remained the main areas for improvement. The resulting prototype fulfills the main goal of the work, because it allows the user to communicate with a VR avatar by voice in both supported languages. Further development could focus on improving the naturalness of the avatar's voice and lip-sync, extending avatar animations, adding spatial context from the virtual environment, or turning the solution into a reusable Unreal Engine plugin.

Acknowledgement

I would like to thank RNDr. Ján Lacko, PhD., for his professional assistance, valuable advice, and consultations during the preparation of this master thesis.

References

[1] Weizenbaum, J. (1966). ELIZA: A computer program for the study of natural language communication between man and machine. Communications of the ACM. vol. 9, no. 1, pp. 36-45, doi: 10.1145/365153.365168.

[2] Johnson, W. L., Rickel, J. (1997). STEVE: An animated pedagogical agent for procedural training in virtual environments. ACM SIGART Bulletin. vol. 8, no. 1-4, pp. 16-21.

[3] Schöbel, S., Schmitt, A., Benner, D., Saqr, M., Janson, A., Leimeister, J. M. (2024). Charting the Evolution and Future of Conversational Agents: A Research Agenda Along Five Waves and New Frontiers. Information Systems Frontiers.

[4] Buldu, K. B., Özdel, S., Lau, K. H. C., Wang, M., Saad, D., Schönborn, S., Boch, A., Kasneci, E., Bozkir, E. (2025). CUIfy the XR: An Open-Source Package to Embed LLM-Powered Conversational Agents in XR. IEEE AIxVR, pp. 192-197, doi: 10.1109/AIxVR63409.2025.00037.

[5] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, doi: 10.1145/3586183.3606763.

[6] Sonlu, S., Bendiksen, B., Durupinar, F., Güdükbay, U. (2025). Effects of Embodiment and Personality in LLM-Based Conversational Agents. IEEE Conference Virtual Reality and 3D User Interfaces, pp. 718-728, doi: 10.1109/VR59515.2025.00094.

[7] Radež, G., Bohak, C. (2024). Integrating environmental awareness into NPCs: contextual conversational interaction in games. CEUR Workshop Proceedings. vol. 3866, pp. 11-29.

[8] Wan, H., Zhang, J., Suria, A. A., Yao, B., Wang, D., Coady, Y., Prpa, M. (2024). Building LLM-based AI Agents in Social Virtual Reality. Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, doi: 10.1145/3613905.3651026.

[9] El Hajji, M., Ait Baha, T., Berka, A., Ait Nacer, H., El Aouifi, H., Es-Saady, Y. (2025). An Architecture for Intelligent Tutoring in Virtual Reality: Integrating LLMs and Multimodal Interaction for Immersive Learning. Information. vol. 16, no. 7, article 556, doi: 10.3390/info16070556.

[10] Christiansen, F. R., Hollensberg, L. N., Jensen, N. B., Julsgaard, K., Jespersen, K. N., Nikolov, I. (2024). Exploring Presence in Interactions with LLM-Driven NPCs: A Comparative Study of Speech Recognition and Dialogue Options. Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology, doi: 10.1145/3641825.3687716.

[11] life-exe (2026). UnrealOpenAIPlugin: Complete Unreal Engine plugin for the OpenAI API. Retrieved online from https://github.com/life-exe/UnrealOpenAIPlugin.

[12] getnamo (2026). Llama-Unreal: Llama.cpp plugin for Unreal Engine 5. Retrieved online from https://github.com/getnamo/Llama-Unreal.

[13] Shiyatzu (2026). OculusLipsyncPlugin-UE5: Oculus LipSync Plugin compiled for Unreal Engine 5. Retrieved online from https://github.com/Shiyatzu/OculusLipsyncPlugin-UE5.

[14] Mozilla (2024). Common Voice Dataset. Retrieved online from https://mozilladatacollective.com/datasets/cmn2e8ojy01l2mm07giwrvaqf.

[15] westbrook (2024). English Accent DataSet. Retrieved online from https://huggingface.co/datasets/westbrook/English_Accent_DataSet.

Author

Bc. Roman Zemaník
Faculty of Informatics
xzemanik@paneurouni.com
Student at the Faculty of Informatics,
currently working as a Frontend Developer.

ISSN: 2453-7497 (online)
ISSN 1338-6468