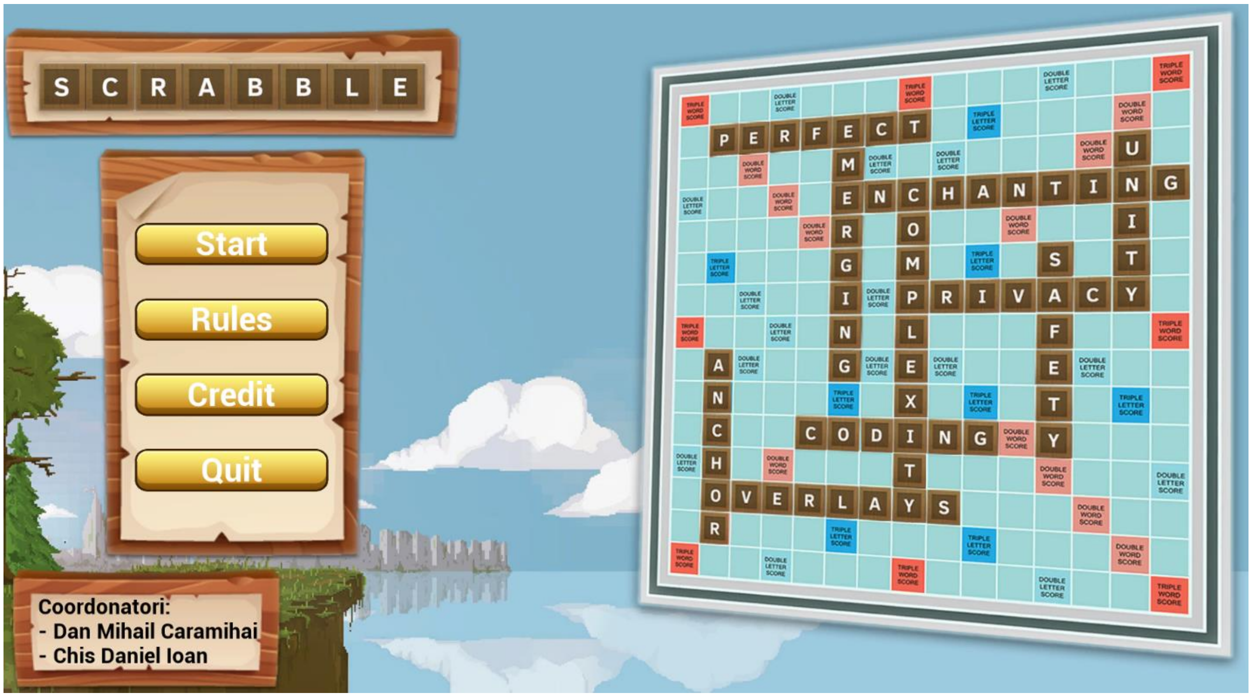
Scrabble Game Report

A Scrabble-based learning platform for acquiring knowledge about accessibility in digital content, as well as virtual reality, augmented reality, and mixed reality.



Project created by:

- Dochia Dimitrie-Razvan (leader)
- Florea Andrei
- Giurescu Adrian-Constantin
- Tachita Bianca-Lucia
- Frumusanu Alexia-Georgiana

Project objectives

The main goal of this project is to develop an interactive and educational Scrabble game designed to facilitate the assimilation of knowledge about accessibility in digital content, as well as emerging technologies such as virtual reality, augmented reality, and mixed reality. The game is designed to encourage active learning by engaging users in a captivating and educational gaming environment. Through this, we aim to combine the traditional elements of classic Scrabble with relevant educational content to offer a unique and interactive learning experience.

Analysis/Documentation (“State of the Art”)

Market Context:

In the initial phase of the project, we conducted an extensive market analysis to identify and evaluate existing Scrabble-like games. This assessment of available digital games provided the foundation for designing a solution that meets the need for interactive education, enhancing the classic Scrabble framework with new and engaging elements.

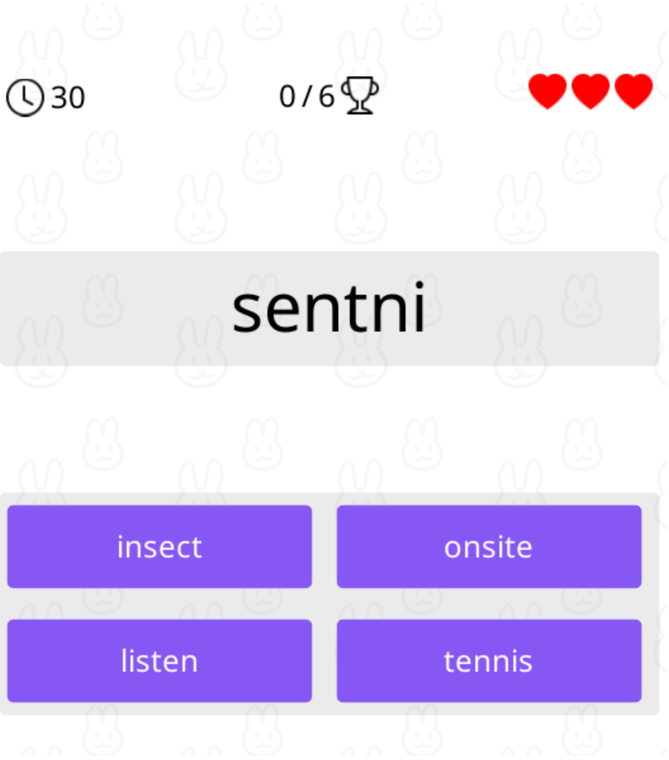
Classic Scrabble Game Dynamics:

In its traditional form, Scrabble engages players in an intellectual challenge of forming words using letter tiles arranged on a board, balancing an extensive vocabulary with strategic point optimization. Our analysis highlighted the strategic and creative importance involved in this process.

Inspiration and Innovation

A key component of our research was identifying and integrating intuitive ways to incorporate questions into the game structure. While studying various iterations of Scrabble, we came across several creative interpretations on the itch.io platform, among which "Tangled Words" by eytau stood out for its unique approach. This game combines mental agility and quick thinking by encouraging word formation from random letters while implementing a dynamic timer and a well-thought-out reward and penalty system.

Image of the game “Tangled Words”:



Gamification and User Experience:

Influenced by the mechanics we discovered, we decided to implement a timer for each question and set score thresholds for level progression, thus enhancing the gamification aspect of the game. Our observations also revealed a tendency to use letter arrangement on the screen as a learning method— a strategy we consider essential for attracting and maintaining user interest.

Visual Quality:

The graphical quality of the analyzed games was also an impressive factor, having a direct impact on the user experience. We aim to achieve high visual standards to complement and enrich the educational process through visual stimulation.

Intermediate Conclusions:

Although many games follow a classic format or deviate significantly from our vision, we have managed to extract valuable elements that have shaped our approach. We have synthesized these findings into a concept that not only pays homage to classic Scrabble but also expands it into a new dimension of learning and interactivity.

Solution Justification

Technology Selection:

The design process for our game began with a detailed analysis of the available technological options. Given our limited prior experience in game development, selecting an appropriate platform was essential. Among the possible options—Godot, Open 3D Engine, Cocos2d, and others—we chose Unreal Engine 5 (UE5) due to its industry popularity, well-structured documentation, and active developer community. The key factors influencing this decision included: Accessibility and ease of learning the Blueprints system, Efficient development through rapid iteration, Intuitive content and resource management, Seamless integration of visual elements, Reduced development time and Strong community support.

Game Concept Design:

A major challenge in designing the game concept was integrating educational questions into the gameplay mechanics. Our chosen methodology was simple yet effective: transforming questions into affirmative sentences with a missing word, which the player must identify and place using letter tiles. This approach not only tests the player’s knowledge but also enhances critical thinking and vocabulary skills.

Example of an Integrated Question:

To illustrate this mechanism, consider the following example:
"One true statement about VR systems is that they use Head- _ _ _ _ _ _ _ Displays to entirely block out the real world and display a virtual environment."
The correct answer is **"MOUNTED"**. Players receive random letter tiles and must form the correct word to fill in the blanks, demonstrating both subject knowledge and linguistic agility.

Game Features:

To ensure an interactive and accessible gameplay experience, we implemented various assistive features:

- CLUE button: Provides hints.
- SKIP button: Allows skipping to the next question without earning points.
- NEXT button: Becomes active after correctly placing the word, allowing progression and score accumulation.
- BOARD button: Displays the Scrabble board, revealing arranged words and tracking progress.

This approach was driven by the desire to balance the complexity of development with the need to create an **appealing and functional final product** within the available time frame. Choosing **UE5** as the development platform and integrating a **question-based system in an interactive format** allowed us to achieve the project's objectives and move forward to the **production phase with confidence and clarity** in our chosen direction.

Implementation Description

Menus:

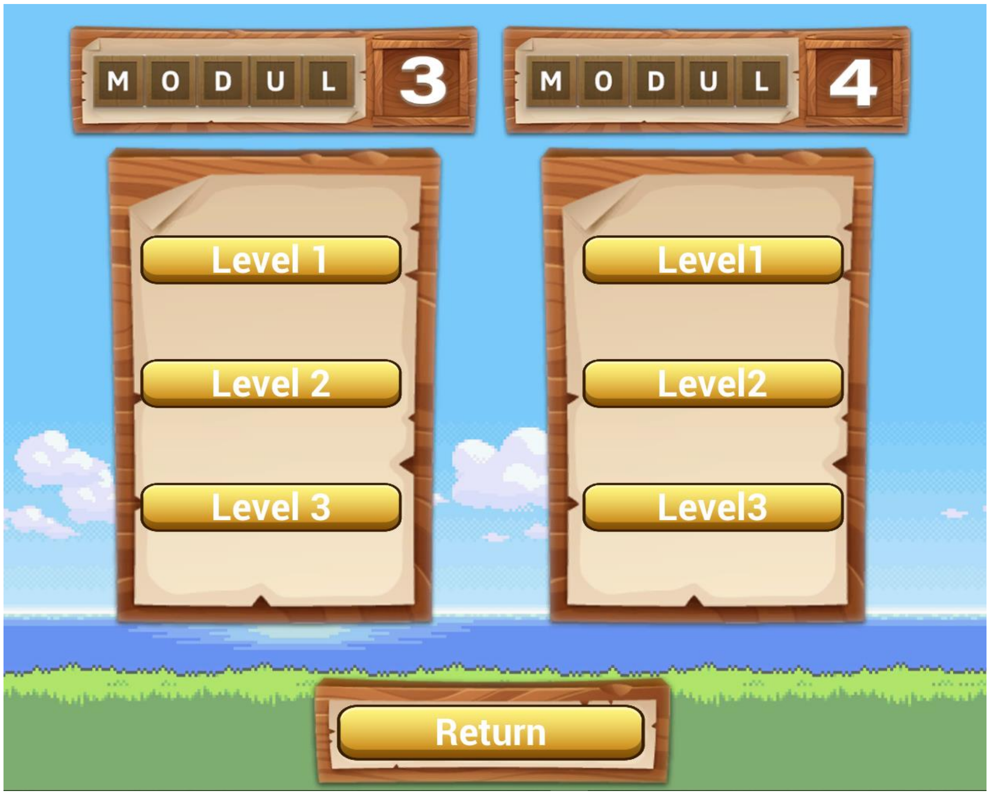
When launching the game, users are greeted by an introductory screen that displays the platform used for development for three seconds, providing immediate recognition of the technology behind the game.

After the introductory sequence, users are directed to the main menu, which serves as the navigation hub of the game. The main menu consists of four essential buttons that help players navigate the application:

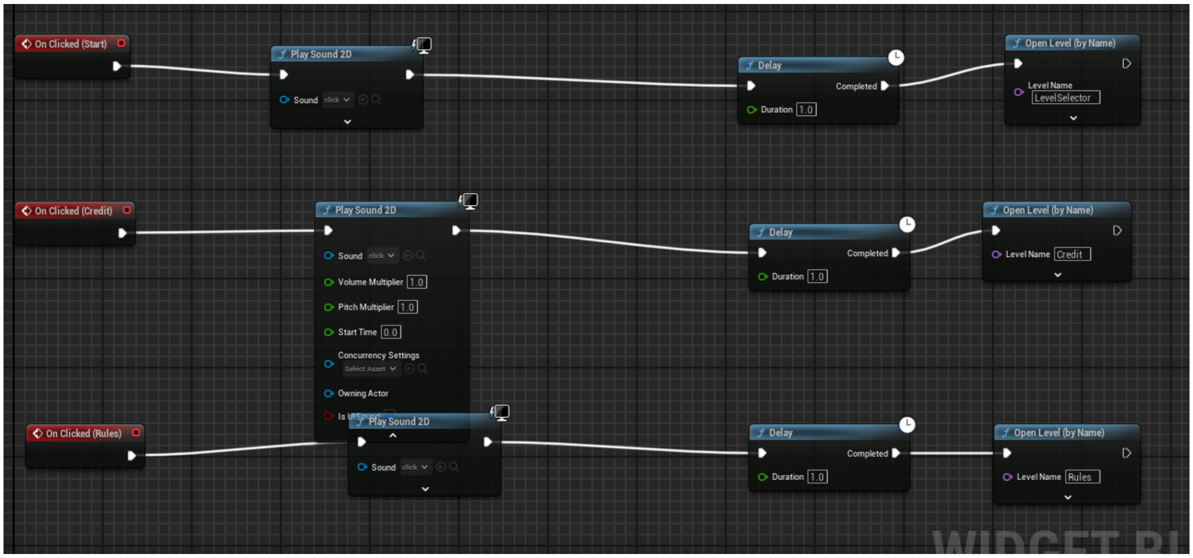
1. START Button: Takes users to the level selection menu.
2. RULES Button: Redirects players to a section where the game rules are explained in detail.
3. CREDITS Button: Displays the development team's credentials.
4. QUIT Button: Allows players to exit the game and close the application.

Moving forward, the level selection menu allows users to choose between different game stages, each corresponding to a learning module. This menu is structured to offer access to three distinct levels per module, with each level focusing on different educational content.

Visual Presentation of the Level Menu:



We designed and implemented the user interface using dedicated Blueprint widgets in Unreal Engine 5. The interface navigation is based on an interactive button system, with each button linked to a specific function and menu screen. By interacting with the buttons, specific commands are triggered, such as the 'Open Level' function, which initiates the opening of the corresponding screen based on the user's selection.



Sound click
Duration 1.0
Level Name LevelSelector

Sound click
Volume Multiplier 1.0
Pitch Multiplier 1.0
Start Time 0.0
Concurrency Settings Select Asset
Owning Actor
Is UISound
Duration 1.0
Level Name Credit

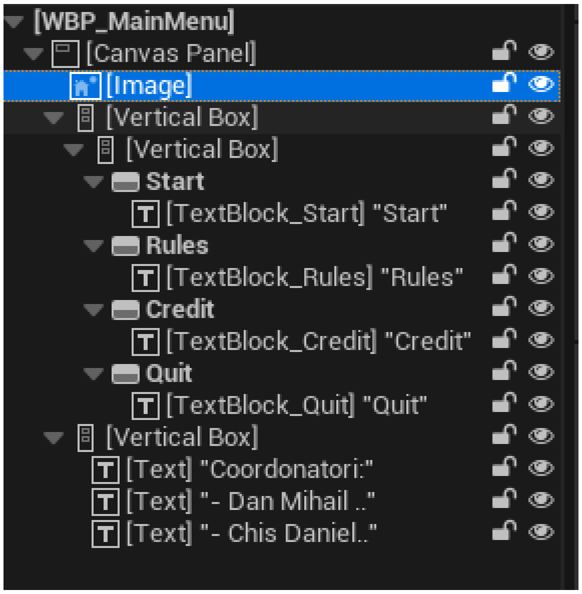
Sound click
Duration 1.0
Level Name Rules

Graphics:

The visual design of the application was created through careful selection and adaptation of images, using the advanced editing capabilities of Adobe Photoshop. This process was guided by the desire to provide a pleasant aesthetic and thematic consistency with the gameplay elements.

Once edited, the images were integrated into the game using the tools available in Unreal Engine 5. We used Blueprint widgets to incorporate graphical elements into the user interface, specifically employing an image widget within the canvas panel, allowing us to display selected textures directly on the game screen.

Image of the canvas panel for the main menu:



```
[WBP_MainMenu]
  [Canvas Panel]
    [Image]
    [Vertical Box]
      [Vertical Box]
        Start
          [TextBlock_Start] "Start"
        Rules
          [TextBlock_Rules] "Rules"
        Credit
          [TextBlock_Credit] "Credit"
        Quit
          [TextBlock_Quit] "Quit"
    [Vertical Box]
      [Text] "Coordonatori:"
      [Text] "- Dan Mihail .."
      [Text] "- Chis Daniel.."
```

Sound:

To enhance the user experience and add an element of immersion to the game, we included a background soundtrack. The music was carefully selected and arranged to complement the game's atmosphere and increase its overall appeal.

User interactions with the interface are accompanied by specific sound effects: a distinct sound is triggered when hovering over buttons, and a different tone plays when clicking on them. This was implemented using the Play Sound 2D function in the widget. This attention to audio details contributes to clear auditory feedback and a dynamic, engaging gaming experience.

Gameplay:

1. Main Game Mechanic:

At the core of the gaming experience is the challenge of completing words using available letters to answer the educational questions presented. Each level starts with a question and a blank space representing the missing word. Users must place the letters in a logical order to form the correct word that fills the blank, thus demonstrating not only their acquired knowledge but also their ability to solve linguistic puzzles under time pressure.

2. Level Flow:

The game contains six different levels, each with its own unique set of questions and answers. The letters required for the answers are provided but are arranged in a random order with each attempt, thereby increasing the level of challenge. Questions are always presented in the same order to allow players to build a strategy as they progress through the levels.

3. Interaction and Feedback:

When a player selects the correct letter, it is automatically placed in the corresponding box on the game board, confirming the correct selection. If an incorrect choice is made, the game provides immediate feedback by displaying a warning message and encourages the player to try again. This immediate response system keeps the game's pace steady and supports the learning process.

4. Time Management:

Each question must be answered within a two-minute window, emphasizing speed and mental agility. If the allotted time for a question expires, the game automatically advances to the next one, with the player actively maintaining the flow and dynamics of the game.

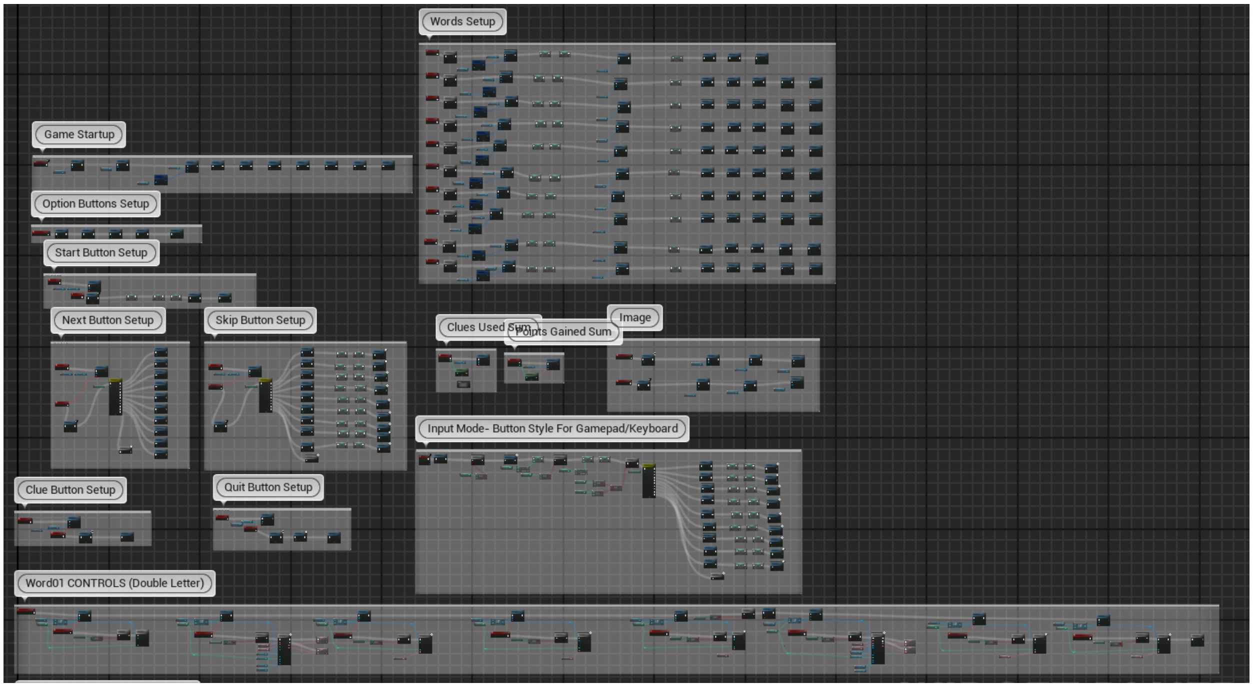
5. In-Game Assistance Elements:

On the game screen, there are four assistive buttons serving different functions:

- NEXT Button: Allows the player to advance to the next question after successfully completing the current word.
- SKIP Button: Provides the option to skip a difficult question without receiving points for that attempt.
- CLUE Button: Helps the player by displaying the next required letter, with a maximum of three clues available.
- BOARD Button: Allows the player to view the Scrabble board to observe how it changes after each answer or omission.

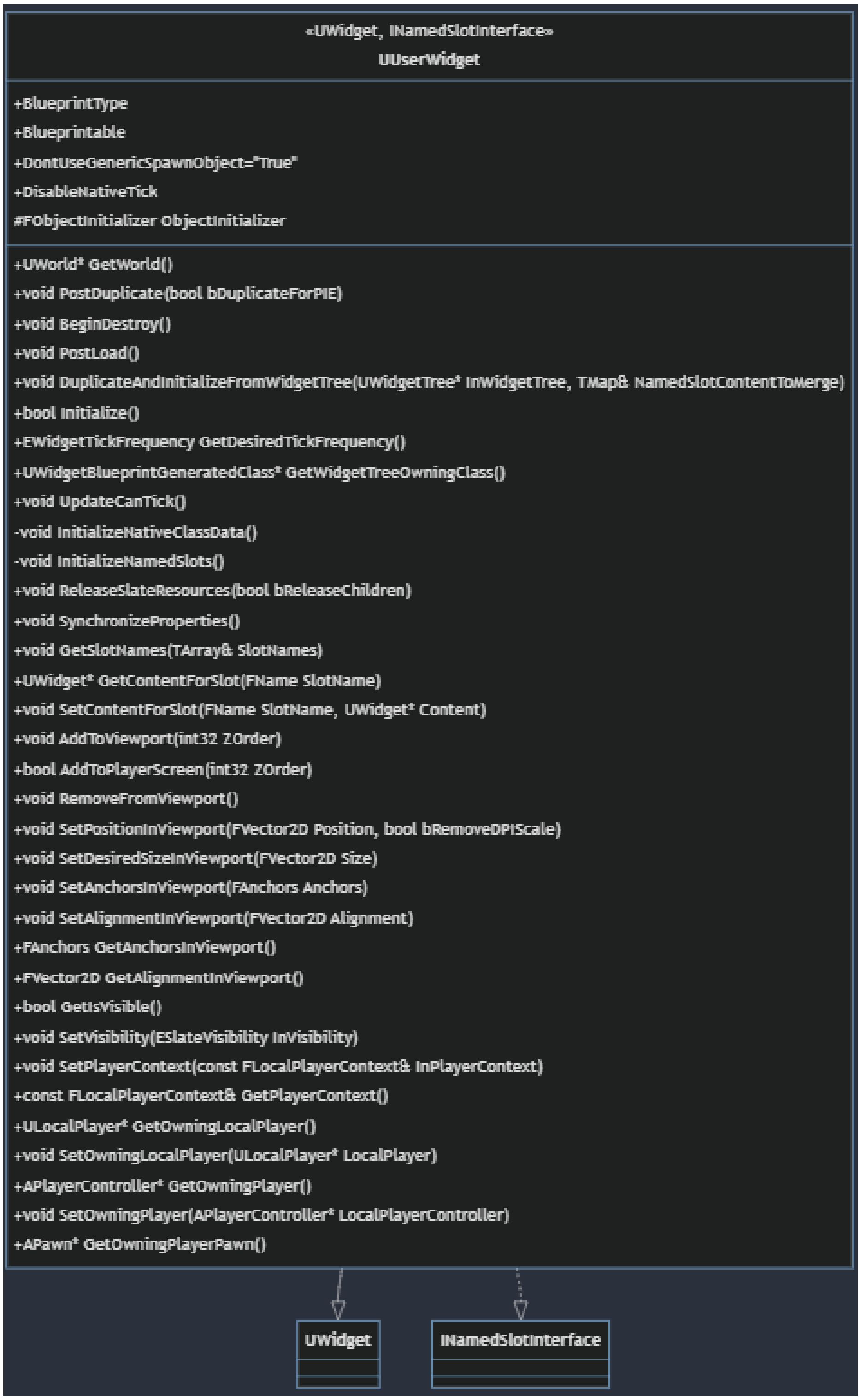
6. Scoring System:

A level is considered won if the player accumulates a score that represents at least 50% of the total possible points. At the end of each level, a summary screen is displayed showing the score obtained and the percentage of the level completed, thus providing a clear overview of the player's performance.



UML Diagrams:

Class Diagram:

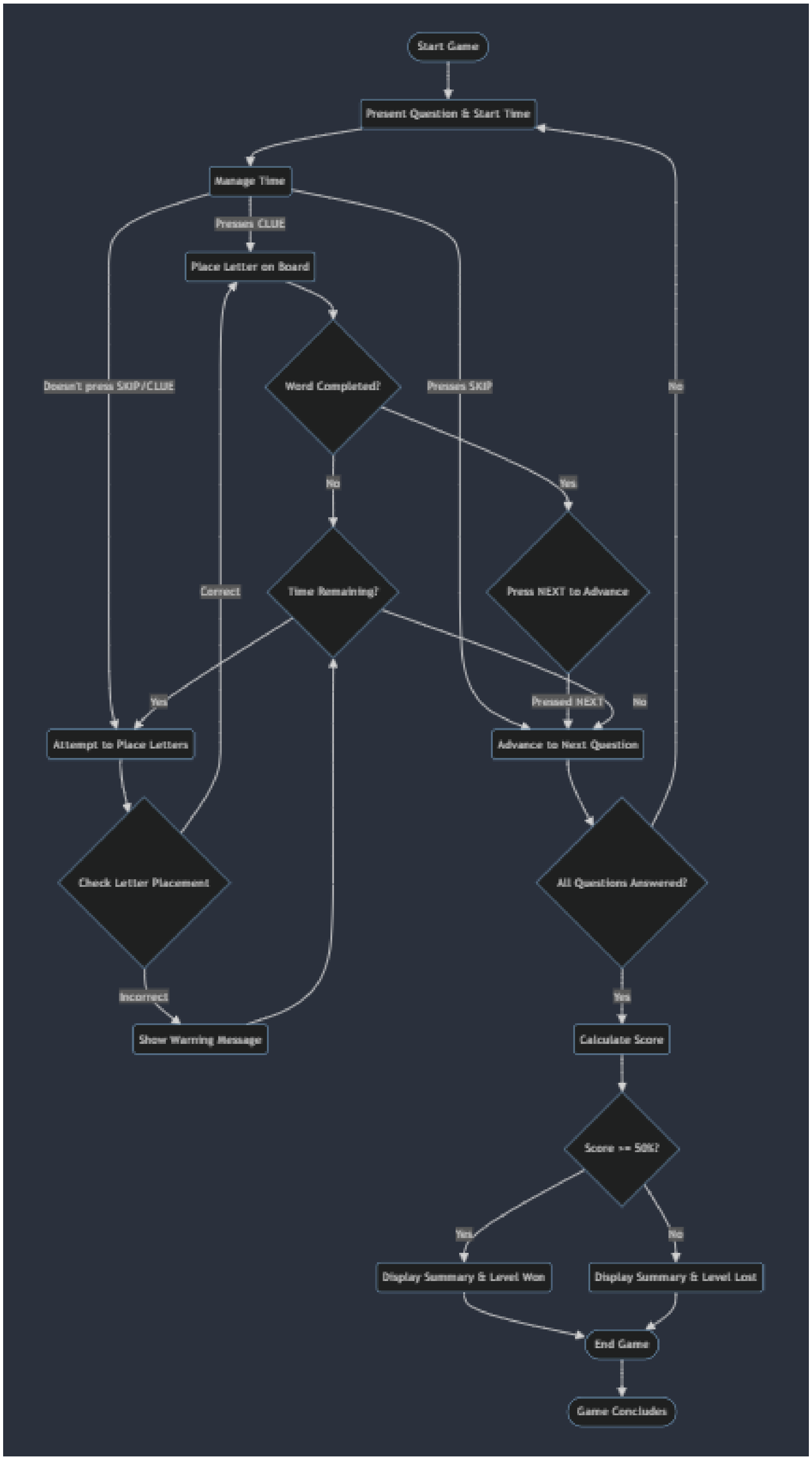


```
«UWidget, INamedSlotInterface»
UUserWidget

+BlueprintType
+Blueprintable
+DontUseGenericSpawnObject="True"
+DisableNativeTick
#FObjectInitializer ObjectInitializer

+UWorld* GetWorld()
+void PostDuplicate(bool bDuplicateForPIE)
+void BeginDestroy()
+void PostLoad()
+void DuplicateAndInitializeFromWidgetTree(UWidgetTree* InWidgetTree, TMap& NamedSlotContentToMerge)
+bool Initialize()
+EWidgetTickFrequency GetDesiredTickFrequency()
+UWidgetBlueprintGeneratedClass* GetWidgetTreeOwningClass()
+void UpdateCanTick()
-void InitializeNativeClassData()
-void InitializeNamedSlots()
+void ReleaseSlateResources(bool bReleaseChildren)
+void SynchronizeProperties()
+void GetSlotNames(TArray& SlotNames)
+UWidget* GetContentForSlot(FName SlotName)
+void SetContentForSlot(FName SlotName, UWidget* Content)
+void AddToViewport(int32 ZOrder)
+bool AddToPlayerScreen(int32 ZOrder)
+void RemoveFromViewport()
+void SetPositionInViewport(FVector2D Position, bool bRemoveDPIScale)
+void SetDesiredSizeInViewport(FVector2D Size)
+void SetAnchorsInViewport(FAnchors Anchors)
+void SetAlignmentInViewport(FVector2D Alignment)
+FAnchors GetAnchorsInViewport()
+FVector2D GetAlignmentInViewport()
+bool GetIsVisible()
+void SetVisibility(ESlateVisibility InVisibility)
+void SetPlayerContext(const FLocalPlayerContext& InPlayerContext)
+const FLocalPlayerContext& GetPlayerContext()
+ULocalPlayer* GetOwningLocalPlayer()
+void SetOwningLocalPlayer(ULocalPlayer* LocalPlayer)
+APlayerController* GetOwningPlayer()
+void SetOwningPlayer(APlayerController* LocalPlayerController)
+APawn* GetOwningPlayerPawn()
```

Activity Diagram:



Clarification of Object-Oriented Programming (OOP) Elements Used in Implementation::

1. Class Example: "UuserWidget" itself is a class, as indicated by the class definition that starts with class UuserWidget.
2. Object Example: Any instance of "UuserWidget," such as UUserWidget myWidget = new UUserWidget();, would be an object of this class.
3. Inheritance: The declaration class UUserWidget : public UWidget, public InamedSlotInterface shows that "UuserWidget" inherits from both "UWidget" and "InamedSlotInterface."
4. Encapsulation: The class contains private or protected data members (such as ColorAndOpacity, ForegroundColor) and public methods to access or modify these (such as getters and setters).
5. Overloading/Overriding: The presence of functions like NativeOnInitialized() in "UuserWidget," which are also present in the base class but with additional or modified functionality in "UuserWidget."
6. Templates: The function CreateWidget<T>() where "T" is a type parameter, represents a template function.

Description of Performance Indicators

Frames per Second (FPS):

The frames per second rate was recorded at 120 FPS, an indicator that evaluates the system's ability to generate and render graphics in one second, symbolizing the visual fluidity of the game.

Response Time (MS):

The response time indicator reached a value of 8.33 milliseconds, representing the time required for the system to process and execute the user's commands, an essential aspect in evaluating the game's responsiveness.

Frame Processing (Frame Time):

To process and display a single frame, the system requires an interval of 8.33 milliseconds, a parameter that determines the overall efficiency of frame-level graphical processing.

Game Execution Time (Game Time):

The game logic execution phase, which includes the time allocated for actions and updates in gameplay, was measured at 4.45 milliseconds per frame.

Draw Time:

The rendering process, which includes drawing graphical elements on the screen, was completed in 3.29 milliseconds, serving as an indicator of the game's graphical performance.

Raster Hardware Interface Test (RHIT Time):

The time allocated for testing the raster hardware interface was recorded at 3.73 milliseconds, highlighting the efficiency of the rasterization component.

GPU Processing Time (GPU Time):

The total processing duration on the graphics processing unit was 6.31 milliseconds, reflecting the time consumed by the GPU to complete graphical tasks in each frame.

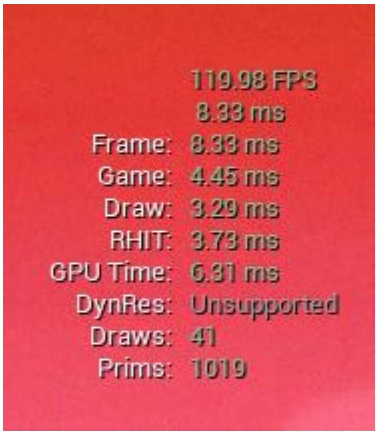
Draw Calls:

A total of 41 draw calls per frame was recorded, indicating the volume of draw commands sent to the GPU and serving as an indicator of the graphical scene's complexity.

Primitives Count:

The number of graphical primitives, such as triangles or rectangles, used in rendering a frame was 1019, providing insight into the geometric complexity of the visual elements in the game.

The above data was extracted directly from Unreal Engine 5:



```
            119.98 FPS
             8.33 ms
    Frame:  8.33 ms
     Game:  4.45 ms
     Draw:  3.29 ms
     RHIT:  3.73 ms
 GPU Time:  6.31 ms
   DynRes:  Unsupported
    Draws:  41
    Prims:  1019
```

Conclusions

Through our project, we successfully achieved the goal of creating an interactive and educational Scrabble game, which successfully blends the classic word game foundation with advanced concepts from the targeted learning modules. Users not only have the opportunity to test and improve their vocabulary and knowledge but are also challenged to think critically and apply their learnings in a dynamic and interactive environment.

Market analysis and prior research were essential in defining our creative and technical direction. Drawing inspiration from games like "Tangled Words," we implemented features that introduce a new gameplay style, such as the timer system and interactive question responses.

Choosing Unreal Engine 5 as our development platform proved to be decisive, allowing us to take full advantage of the Blueprints system for quick and efficient implementation of our ideas.

From a graphical perspective, we achieved a high level of visual excellence, and in terms of audio, we succeeded in creating a pleasant atmosphere that enhances the user experience. In terms of performance, our game runs at high standards, with fast response times and a frame rate that ensures a smooth gameplay experience.

The feedback received from the second team was positive, confirming that we met our goal of providing a valuable educational experience while maintaining player engagement and enjoyment.

Ultimately, we found that all previously established targets and objectives were successfully and efficiently met.