Text

Description automatically generated

Computer Games Development CW208

Technical Design Document

Year IV

|  |
| --- |
|  |

Davids Jalisevs

C00239534

28/04/

Contents

[UML Diagrams 3](#_Toc133353902)

[**Simple Game Loop Diagram** 3](#_Toc133353904)

[Anwil Works(sending game data to the server) 4](#_Toc133353905)

**Benefits of collecting game data**………………………………………………………….….**7**

Steam VR…………………………………………………………………………………….8

Steam VR Unity Plug in………………………………………………………………………9

Unity Game engine…………………………………………………………………………...10

Controller haptics…………………………………………………………………………….11

CRC Cards…….……………………………………………………………………………..12

References……………………………………………………………………………………16

# UML Diagrams

## Overview Diagram

Chart, diagram, box and whisker chart

Description automatically generated

## **Simple Game Loop Diagram**

Diagram

Description automatically generated

# Anvil Works

Anvil Works is a web service that provides a cloud-based development environment for building and deploying web applications quickly. It offers a range of tools, including a drag-and-drop interface for building user interfaces, database integration, and serverless functions.

Anvil's drag-and-drop interface allows developers to build user interfaces quickly, without needing to write code. This can save time and effort, especially for developers who are not experienced in web development.

One of the most beneficial features of Anvil is its seamless integration with PostgreSQL, a widely used open-source relational database management system. This enables you to store and analyse data in a centralized location, making it easy to monitor game data and make informed decisions about your game's design and user experience.

code demonstrates a straightforward way to send data from your game to the Anvil server using the UnityWebRequest class. The data is sent in JSON format to the URL specified in the 'url' variable. This allows to store game data in a structured manner and analyse it using the tools provided by Anvil.

Text

Description automatically generated

Then calling this function at any time in the game would execute the postData

Text

Description automatically generated

Table data in anvil server

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, chart, bar chart

Description automatically generated

WEBSITE TO CHECK collected data - https://c00239534-analysis.anvil.app/

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Benefits of collecting game data**

By monitoring game data, you can gain insights into user behaviour, game analytics, and other metrics. This can help you make informed decisions about the design and user experience of your game, ultimately leading to a more polished and professional-looking product.

By collecting data on how players interact with the game, developers can gain insights into which aspects of the game are working well and which need improvement. For example, data on player engagement can help identify which levels or game mechanics are most popular, allowing developers to focus on improving those areas. Similarly, data on player retention can help identify areas where players are dropping off, allowing developers to adjust improve the overall experience.

In addition to its benefits for game design, data can also be used for marketing and monetization purposes. By analyzing data on player behavior and preferences, developers can tailor advertising and promotional campaigns to better reach their target audience. Similarly, data on player spending habits can help developers optimize monetization strategies, such as in-game purchases or subscription models.

The use of Anvil Works as a web service provides a powerful toolset for game developers, allowing them to easily monitor game data and make informed decisions about their game's design and user experience. Your code for sending data to the Anvil server demonstrates a simple and effective way to integrate this web service into your game.

Chart, histogram, waterfall chart

Description automatically generated

**SteamVR**

SteamVR is a virtual reality (VR) platform developed by Valve Corporation that is designed to work with a variety of VR headsets. It provides a set of tools and APIs that developers can use to create VR applications that run on the Steam platform. SteamVR supports a wide range of VR hardware, including the HTC Vive, Oculus Rift, Windows Mixed Reality headsets, and Valve's own Index headset.

One of the key features of SteamVR is its room-scale tracking system, which uses a combination of base stations and sensors on the VR headset to track the user's movement in physical space. This allows for a more immersive VR experience, as users can move around freely within the virtual environment.

SteamVR also provides a range of input options for VR applications, including hand-held controllers and motion-tracked devices like the Vive Tracker. These input options allow developers to create more interactive and engaging VR experiences, as users can manipulate virtual objects with their hands or other physical devices.

In addition to its hardware support and input options, SteamVR also provides a range of tools and APIs for developers to use in creating VR applications. These include tools for rendering 3D graphics in VR, managing input from VR devices, and creating user interfaces for VR applications.

One of the most useful tools provided by SteamVR is its OpenVR SDK, which is an open-source set of APIs that allow developers to create VR applications that can run on a variety of VR hardware. This makes it easier for developers to create VR applications that can reach a wider audience, as they don't have to develop specifically for a single VR headset.

SteamVR also provides a range of community resources for VR developers, including forums, tutorials, and documentation. These resources can be invaluable for developers who are just getting started with VR development or who are looking for help with specific technical issues.

**SteamVR unity plugin**

One of the key features of the SteamVR Unity Plugin is its support for a wide range of VR hardware, including the HTC Vive, Oculus Rift, and Valve's own Index headset. This makes it easier for developers to create VR applications that can reach a wider audience, as they don't have to develop specifically for a single VR headset.

The SteamVR Unity Plugin also provides a range of input options for VR applications, including hand-held controllers and motion-tracked devices like the Vive Tracker. These input options allow developers to create more interactive and engaging VR experiences, as users can manipulate virtual objects with their hands or other physical devices.

In addition to its hardware support and input options, the SteamVR Unity Plugin also provides a range of tools and APIs for developers to use in creating VR applications. These include tools for rendering 3D graphics in VR, managing input from VR devices, and creating user interfaces for VR applications.

One of the most useful tools provided by the SteamVR Unity Plugin is its prefab system, which allows developers to easily add VR functionality to their Unity projects. The plugin provides a range of prefabs for common VR interactions, such as teleportation and object manipulation, that developers can easily drop into their scenes.

The SteamVR Unity Plugin also includes support for SteamVR Input, which is a system that allows developers to define and customize how users interact with their VR applications. This system provides a flexible and powerful way for developers to create custom input mappings and control schemes for their VR applications.

SteamVR Unity Plugin provides a range of community resources for Unity developers, including forums, tutorials, and documentation. These resources can be invaluable for developers who are just getting started with VR development or who are looking for help with specific technical issues.

SteamVR Unity Plugin is a very powerful tool for Unity developers who are looking to create VR applications that run on the SteamVR platform.

**Unity Game Engine**

Unity is a popular game engine that is widely used by game developers to create 2D and 3D games across multiple platforms, including PC, mobile devices, consoles, and even virtual and augmented reality (VR/AR) systems.

One of the key strengths of Unity is its accessibility, as it provides a user-friendly interface and a wealth of resources and documentation to help developers get started quickly. This makes it an ideal choice for developers who may be new to game development or who are looking to prototype and test ideas quickly.

Unity's flexible and powerful scripting system is another key feature that makes it popular among game developers. Unity supports multiple programming languages, including C#, JavaScript, and Boo, and provides a range of tools and APIs for working with game objects, physics, animations, and more. This allows developers to create complex and interactive game systems that can be easily controlled and modified through code.

Another strength of Unity is its cross-platform support, as it allows developers to create games that can run on multiple platforms with minimal changes. This can save time and effort in the development process, as developers can focus on creating a single game that can reach a wide audience.

Unity also provides a range of tools and features for creating high-quality graphics and visual effects, including a built-in particle system, support for real-time lighting and shadows, and integration with third-party tools like Adobe Photoshop and Autodesk Maya. This allows developers to create visually stunning games with ease.

Unity provides a wealth of community resources, including forums, tutorials, and documentation, which can be invaluable for developers who are just getting started with the engine or who are looking for help with specific technical issues.

Unity is a powerful game engine that provides developers with the tools and resources they need to create high-quality games across multiple platforms. Whether you're a seasoned game developer or just starting out, Unity is definitely worth considering for your next game project.

**Controller Haptics**

Haptics are an essential part of creating a fully immersive VR game experience. Haptic feedback allows players to feel as though they are physically interacting with objects in the virtual environment, adding an extra layer of realism to the game. Haptics can include vibrations, forces, and other sensations that simulate the feeling of touch, such as the sensation of hitting a wall or the resistance of pulling a lever.

There are several different types of haptic feedback that can be used in VR games, including hand-held controllers, wearable haptic vests or suits, and even integrated haptic feedback in the VR headset itself. By incorporating haptic feedback into your game, you can enhance the sense of presence and immersion that players experience.

One of the key benefits of haptic feedback in VR games is that it can improve player performance and engagement. By providing feedback that is aligned with the actions players are taking, haptics can help players feel more connected to the virtual environment and better understand how they are interacting with it. This can lead to improved hand-eye coordination, faster reaction times, and a more engaging overall gameplay experience.

Another important benefit of haptic feedback is that it can help to mitigate motion sickness in VR. By providing physical feedback that aligns with the movement of the virtual environment, haptics can help to ground players and reduce the feeling of disorientation that can sometimes occur in VR.

Overall, incorporating haptic feedback into VR game can greatly enhance the sense of immersion and engagement for players, and is an important consideration when designing and developing your game. By carefully designing and implementing haptic feedback, you can create a more realistic and engaging VR game experience that will keep players coming back for more.

**CRC Cards**

|  |  |
| --- | --- |
| PlayerController | |
| Responsibilities | Collaborators |
| * Controls Player Movement |  |

|  |  |
| --- | --- |
| FlyingEnemy | |
| Responsibilities | Collaborators |
| * Controls flying enemy states * Move flying enemy * Shoot bullets at Npc * Shoot bullets at the player * Destroy buildings * Control health state of the enemy | * gameManager * flyingSpawner * playerController |

|  |  |
| --- | --- |
| GreenDogAlien | |
| Responsibilities | Collaborators |
| * Walk around in the designated area * Kill any npc coming close * Kill player coming close * Control animations * Destroy buildings * Control health state of the enemy * Create patrol destinations | * gameManager * flyingSpawner * playerController * healthManager |

|  |  |
| --- | --- |
| AlienNPC | |
| Responsibilities | Collaborators |
| * Walk around until player is located * Chase the player once located * Kill player * Kill NPCs * Control movement of the enemy * Control health state of the enemy | * gameManager * flyingSpawner * playerController * healthManager |

|  |  |
| --- | --- |
| ProjectileThrower | |
| Responsibilities | Collaborators |
| * Find closest enemy with correct tags * Start shooting projectile once in the range * Time the projectiles | * walkingNPCs * player * flyingEnemy |

|  |  |
| --- | --- |
| Bulletscript | |
| Responsibilities | Collaborators |
| * check colliders of the projectile | * walkingNPCs * player * healthManager |

|  |  |
| --- | --- |
| FireBallScript | |
| Responsibilities | Collaborators |
| * Shooting the fire ball * Moving the fire ball * Destroying fire ball | * player |

|  |  |
| --- | --- |
| flyingSpawner | |
| Responsibilities | Collaborators |
| * Spawn and initiate flying enemy * Create waves of spawn for flying enemy * Count waves of flying enemy * Start next wave after wave. * Display wave text announcement | * FlyingEnemy |

|  |  |
| --- | --- |
| DestroyBuildings | |
| Responsibilities | Collaborators |
| * Check for the health of the buildings * Destroy building once ran out of health * Spawn small buildings in the place of big building to create destruction effect | * Projectile * GreenAlienDog * Fireball * RedDog |

|  |  |
| --- | --- |
| PostToServer | |
| Responsibilities | Collaborators |
| * Get the url of server * Set http method to POST * Send data to server | * GameManager |

|  |  |
| --- | --- |
| RedDogSpawner | |
| Responsibilities | Collaborators |
| * Spawn and initiate dog enemy * Count current amount of dogs * Make sure no more than max dogs exist | * AlienRedDog |

|  |  |
| --- | --- |
| GreenAlienSpawner | |
| Responsibilities | Collaborators |
| * Spawn and initiate alien enemy * Count current amount of aliens * Make sure no more than max alien exist | * GreenAlien |

|  |  |
| --- | --- |
| GameManager | |
| Responsibilities | Collaborators |
| * Display all the text in the game * Send data to the server * Count all the data | * FlyingEnemy * Fireball * Spawners * HealthManager |

|  |  |
| --- | --- |
| HealthManager | |
| Responsibilities | Collaborators |
| * Have a health for the player * Deduct the health of the player * Animate health bar with the amount of health | * player |

**References**

Bui, T., Peck, T., & Rizzo, A. S. (2019). The effects of telekinesis on presence, enjoyment, and engagement in virtual reality games. Entertainment Computing, 29, 1-11. <https://doi.org/10.1016/j.entcom.2018.10.002>

Kim, S., Han, J. H., Choi, S., & Lee, J. (2020). A comparison of three game engines for virtual reality development. Multimedia Tools and Applications, 79(13), 8517-8542. <https://doi.org/10.1007/s11042-019-07945-1>

Kim, Y., Lee, J., Lee, D., Lee, M., Kim, D., Lee, Y., & Kim, J. (2018). A comparative study of game immersion using virtual reality and non-virtual reality displays. Journal of the Korea Game Society, 18(5), 91-100. <https://doi.org/10.7587/jkgs.2018.18.5.91>

Ma, X., Fan, Y., Chen, G., Xie, H., & Guo, Z. (2019). Effect of superpowers on player engagement in VR games. Journal of Visual Languages and Computing, 53, 26-33. <https://doi.org/10.1016/j.jvlc.2018.11.002>

Schwind, V., Koenig, S., & Kuhlen, T. (2019). Comparing VR immersion in a head-mounted versus a CAVE display. Virtual Reality, 23(4), 375-386. <https://doi.org/10.1007/s10055-019-00370-5>

Szymon Piskorz – Previous years Student TDD document

https://cloud.smartdraw.com/ - UML diagrams

Anvil.Works