MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Game Engineering and Simulation Technology

Modular game engine in Rust - Comparing performance and memory usage of subsystems to C++

By: Lukas Vogl, BSc.

Student Number: gs16m007

Supervisors: Dipl.-Ing. Stefan Reinalter
Mag.rer.nat. Dr.techn. Eugen Jiresch

Wien, April 16, 2018

FH University of Applied Sciences TECHNIKUM WIEN

Declaration

Wien, April 16, 2018 Signature

Kurzfassung

Modulare Spieleengines zeichnen sich dadurch aus, dass sie intern aus verschiedenen Subsystemen bestehen die unterschiedlichste Aufgaben abarbeiten. Beispielhafte Systeme sind unter anderem Speichermanagement, Rendering oder Physiksimulation. Die Gemeinsamkeit zwischen den Systemen, unabhägig davon wie hardwarenahe oder abstrakt diese sind, sind Aspekte wie Performance und Speicherverbrauch. Um möglichst viel Kontrolle über diese Bereiche zu haben entscheiden sich viele EntwicklerInnen für Systemprogrammiersprachen wie C++ als Entwicklungswerkzeug. Im Zuge dieser Arbeit wird der Autor die seit 2015 existierende Programmiersprache Rust verwenden um ausgewählte Subsysteme einer modularen Spieleengine zu implementieren. Ziel der Arbeit ist es zu untersuchen, ob Rust durch seine neuen Konzepte gängige Schwierigkeiten bei der C++ Entwicklung vermeiden und gleichzeitig eine gleichwertige Performance liefern kann. Dafür werden die in Rust implementierten Systeme zusätzlich in C++ implementiert und anschließend in verschiedenen Szenarien vermessen und verglichen. Aus den Ergebnissen wird evaluiert ob Rust als Programmiersprache für Spieleengines in Frage kommt. Zusätzlich werden die Implementierungsdetails der verschiedenen Sprachen und Systeme behandelt, wodruch aufgezeigt wird welche Unterschiede zwischen den beiden Sprachen bestehen.

Schlagworte: Rust, C++, Engine, Speichermanagement, Performance

Abstract

Modular game engines are defined by the fact that they are composed of different subsystems working on many distinct tasks. Exemplary systems are, inter alia, memory management, rendering or physics simulation. The similarity between the systems, regardless of how low-level or abstract they are, are performance and memory consumption. To gain control over these fields most programmers choose system programming languages such as C++ as development tool. In this thesis the the author chose the programming language Rust to implement selected subsystems of a modular game engine. Is it the goal of the thesis to investigate whether Rust can avoid common difficulties known from C++ due to its new concepts while maintaining C++ like performance. For this purpose the selected systems will also be implemented in C++. They are then surveyed in different scenarios and compared to each other. The results are evaluated to see whether it is worth considering using Rust as a language for game engine programming. Furthermore the implementation details ofthe different languages and systems are discussed whereby the differences between the two languages are outlined;

Keywords: Rust, C++, Engine, Memory management, performance

Acknowledgements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Introduction **1**

2 Game engine architecture **2**

2.1 Evolution of game engines 2
2.2 Modern commercial game engines 3
2.2.1 Unreal Engine 4 4
2.2.2 Unity 6
2.2.3 Molecular 9
2.2.4 Tombstone 9
2.3 Rust game engines 10
2.3.1 Piston 10
2.3.2 Amethyst 11
2.4 Tools and asset pipeline 11
2.4.1 Editor 11
2.4.2 Asset pipeline 11
2.5 Engine subsystems overview 12
2.5.1 Memory Management 12
2.5.2 Job System 12
2.5.3 Rendering 13
2.5.4 Entity Component System 13
2.5.5 Scripting 13

3 Rust **14**

3.1 Current state 14
3.2 Rust ecosystem 14
3.3 Concepts 14
3.3.1 Borrow checker 15
3.3.2 Traits 15
3.3.3 Hygienic macros 15
3.4 Pitfalls 15

4 Subsystem implementation **16**

4.1 Spark engine architecture 16
4.2 Development environment 16
4.3 Rendering framework 17

4.4 Memory Management 17
4.4.1 API 17
4.4.2 C++ implementation 17
4.4.3 Rust implementation 18
4.5 Containers 18
4.5.1 API 18
4.5.2 C++ implementation 18
4.5.3 Rust implementation 19
4.6 Entity Component System 19
4.6.1 API 19
4.6.2 C++ implementation 19
4.6.3 Rust implementation 20

5 Measures & Comparisons 20
5.1 Environment 20
5.2 Testcases 20
5.3 C++ performance 21
5.3.1 Memory Management 21
5.3.2 Container 21
5.3.3 Entity Component System 21
5.4 Rust performance 22
5.4.1 Memory Management 22
5.4.2 Container 22
5.4.3 Entity Component System 22
5.5 Comparison 23

6 Conclusion 23

Bibliography 24

List of Figures 25

List of Tables 26

List of Code 27

List of Abbreviations 28

1 Introduction

Game engines are an essential part of the gaming industry. Todays state-of-the-art game engines have committed themselves to the goal of creating visually appealing games while providing reasonable performance. Achieving this requires the engine engineers to invest a great amount of time and know-how of underlying hardware. Many of these engines, choosing Unity and Unreal Engine 4 as example, are using C++ as underlying technology. C++ is the language of choice due to it's capabilities of managing memory manually without the limitations of a garbage collector. These capabilities are the foundation for high performance software and essential to game engines. But while the benefits of manual memory management are indisputable it also comes with common pitfalls.

This thesis aims to examine whether the system programming language Rust can be used as a replacement for C++. Rust claims to avoid pitfalls made in C++ while maintaining similar performance. As a basis for discussion the author will implement selected engine subsystems: memory management, containers and an Entity Component System (ECS). All systems, except for the ECS which already exists in Rust, are written in Rust and C++ to later measure and compare their performance in different scenarios. The results of the measurements shall then serve as the basis for the discussion if Rust can be considered as a viable language for game engine programming.

Chapter 2 will introduce the reader to the history and evolution of game engines. It will also outline state-of-the-art products and shortly describe them. Furthermore, it introduces important tools that tightly related to the underlying engine and the concept of an asset pipeline, concluding the chapter with a section presenting the theory and examples of selected engine subsystems. The next chapter provides the reader with an overview of the Rust programming language. It describes the current state of Rust and creates a basic understanding of it by introducing the most important concepts and patterns. It will then compare common and well-known C++ problems and pitfalls with corresponding code in Rust. At the end the author outlines encountered difficulties that can occur when working with Rust. In chapter 4 the author talks about the implementation details of the implemented subsystems and where the differences between Rust and C++ are visible. The development process, architecture and project setup of the Spark engine (the implemented submodules will serve as a basis for this engine in future work) will be discussed. The performance measurement results and observed scenarios are then compared and discussed in chapter 5. Chapter 6 will then finish the thesis with the conclusion.

2 Game engine architecture

Game engines are tightly connected to the evolution of video games themselves. Where 50 years ago a game was build out of hardware the rapid development of a computer's processing power and storage capabilities changed the process of how games are made. Today a game runs on machines assembled from multiple cores, several Gigabytes (GBs) of Random-Access-Memory (RAM) and a powerful graphics processing unit (GPU). But whether the target platform is a PC or a specialized gaming console every modern game has to fulfill certain constraints to be a viable product. This chapter will highlight some of the most important milestones in the history of video games and game engines. It will also give an overview of well-known products on the game engine market and will conclude with the description of selected submodules, the underlying building blocks of a game engine.

2.1 Evolution of game engines

When the first developers started to create video games, the term *game engine* was non existent. At this time the software that ran the simulation a player experienced was tailored to the needs of a specific genre, hardware and game. It was then in the 1990s and with the rise of games like *Doom (1993)* and *Quake (1996)* that certain software was referred to as a *game engine*. The mentioned games separated their technical backbones into different components, creating an architecture that distinguishes between core software modules and game specific entities such as art assets, levels and the general rules of the game. Due to the well-designed architecture and separations the effort to create a new game, where the general concept is the same, was reduced from writing every system and piece of code to creating new art and only tweak and configure the software of previous games. This was also the birth of the *modding* community, where individuals and also studios modify existing games or engine software to create new content or whole games.

From that time on the developers created their games with modding and future extension in mind. Smaller studios started to license the parts of the engine software they could not afford to create by themselves, be it money, time or man power. With the concept of licensing studios, that created the extensible and reusable software packages, created another source of income. But while the goal of an extensible and reusable software collection is a desirable one, the line between a game and an engine is often softer than desired. Because of the nature of games and their specific genre rules it is hard, if not even impossible, to develop a generic engine that can serve as a template for every game. It became the responsibility of the engine developer to find the balance between general-purpose functionality and game or platform tailored optimizations. This trade-off has to be made because the developer can only assume how the software will be used. And so a game engine developed and optimized for rendering first person shooters (FPSs) will probably not run a real-time strategy (RTS) game with maximum performance due to the different rule sets and features both genres require.

Empowered by the wish of creating games that can be modded and licensed bigger studios started to create commercial engines. *Id Software*, the company that created *Doom* and the *Quake* trilogy, opened up the field with their FPS engines in the early 1990s. *Id Software* was then followed by *Epic Games, Inc.*, the creator of the Unreal Engine, which served as the basis for their well-known game *Unreal* and later the *Unreal Tournament* series. The current version, the Unreal Engine 4, is one of the most popular game engines of our time and will be described in the next section. Other game engines started that time and which should not be left unmentioned are *CryENGINE* (Crytech), *Source Engine* (Valve), *Frostbite* (DICE) and *Unity*.

A long time many, if not all, of the named engines followed a model of selling licenses to developers for accessing the engine and the source code. But it was around 2009 and again 2015 that big engine developers including Unity and Epic Games, decided to rework their business model and let developers use their software for free. The license terms often include a revenue share if the game should be successful but the basic usage of the engines is free most of the time. Although this certainly had a huge impact on smaller engines and teams that cannot compete with the teams working at Epic or Unity, this decision lowered the entrance barrier for ongoing game developers and students and can be seen as one of the biggest milestones in the modern history of game engines.

Speaking of Unity and Epic, the next section will describe modern game engines in the market, how they work and what they are used for. In contrast to Unity and Unreal Engine 4, two smaller engines will be described to show the difference between the approaches and why the flexibility of smaller engines and teams can also be an advantage over big software projects.

2.2 Modern commercial game engines

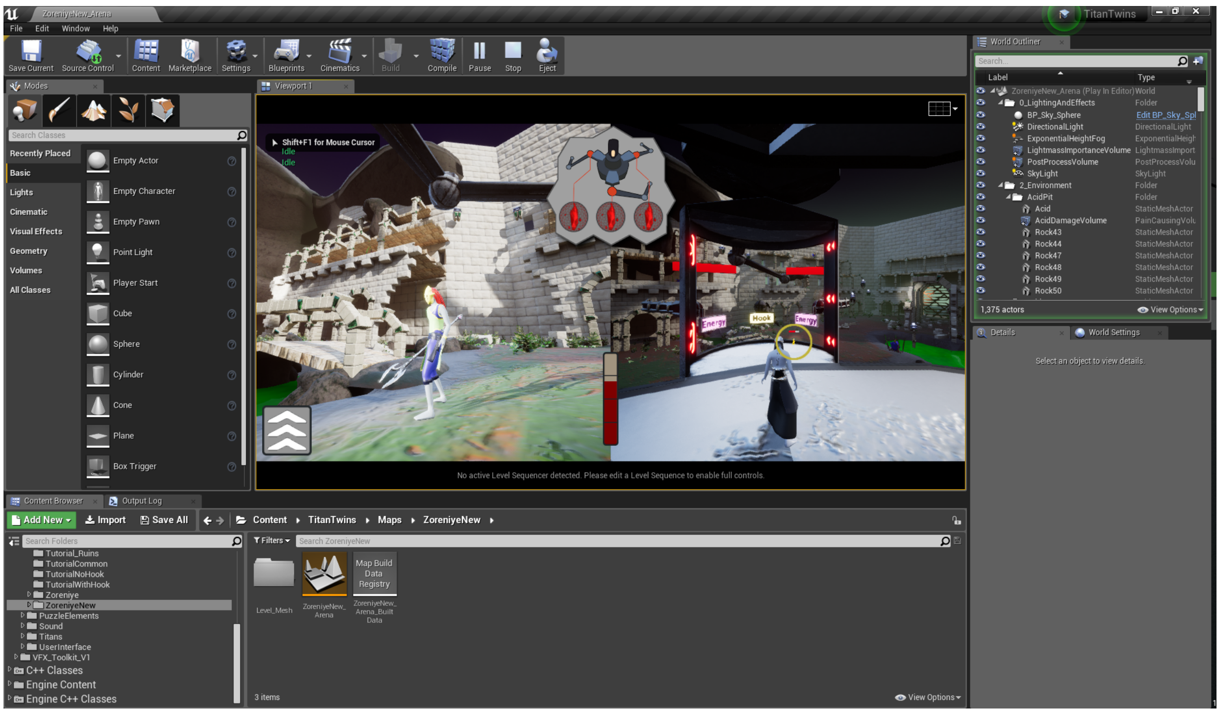
As described in the previous section game engines evolved from extensible game or genre tailored software to standalone viable products used to create different games. The market is actually dominated by two big engines, Unity and Unreal Engine 4, whereas the rest is separated between custom in-house and several medium to small engines and open source projects. The author is going to describe the features and license models of the two big solutions as well as of two smaller but more flexible ones.

2.2.1 Unreal Engine 4

As already known *Epic Games, Inc.* released the first version of the Unreal Engine with their game *Unreal*. The engine was quickly adopted and Epic developed two more generations, Unreal Engine 2 and 3, before the current generation, called Unreal Engine 4 (UE4), was released. All generations powered well-known games such as Deus Ex (UE1), Tom Clancy's Splinter Cell: Pandora Tomorrow (UE2), Gears of War (UE3) or Fortnite (UE4), to name a few.

While previous generations could be licensed by developers to power their products, Epic first changed their licensing model to a monthly fee and alter lowered the access barrier even more by getting rid of any license fee at all. The current version of the UE4 is for free and source code access can be requested at github (https://github.com/EpicGames/UnrealEngine). This move allowed many developers to create their projects with UE4 and the developer has only to pay Epic a 5% revenue share if the game makes more than 3000$ per calendar quarter in profit.

When working with UE4 the developer can choose to build everything from source or work with distributed pre-built binaries. Due to the fact that the Unreal family of engines was developed with first- and third person shooters in mind, the source code access is often appreciated by developers to tweak and rework the engine in a way to better run games from other genres. Regardless whether the engine was built from source or not when working with it the user interacts with the integrated Editor, often also referred to as *UnrealEd*. The editor serves as the user interface (UI) for the many tools available.



Label	Type
ZoreniyeNew_Arena (Play In Editor)	World
0_LightingAndEffects	Folder
BP_Sky_Sphere	Edit BP_Sky_Spl
DirectionalLight	DirectionalLight
ExponentialHeightFog	ExponentialHeigh
LightmassImportanceVolume	LightmassImport
PostProcessVolume	PostProcessVolu
SkyLight	SkyLight
2_Environment	Folder
AcidPit	Folder
Acid	StaticMeshActor
AcidDamageVolume	PainCausingVolu
Rock43	StaticMeshActor
Rock44	StaticMeshActor
Rock47	StaticMeshActor
Rock48	StaticMeshActor
Rock49	StaticMeshActor
Rock50	StaticMeshActor

1,375 actors

Figure 1: The editor shipped with UE4

Going from left to right and top to bottom, Figure 1 shows a selection for actors (objects that can be placed in the game world), a game world view, the world outliner (hierarchy of actors) and the content browser, that lists all folders and contained assets. The editor is a place where designers can create the game world, where artists can author special effects and their assets directly in the game and where designers and programmers can develop the rules and logic needed to drive the world. For creating this logic UE4 offers two possible ways: scripting via the *Blueprint* system or directly in C++. UE4 comes with the integrated *Blueprint Visual Scripting* system that allows for gameplay programming using the concept of a node-based graph. This system is versatile and if it has reached it's capacities a programmer can still implemented the necessary or performance critical parts in C++ and expose an interface for the designer to use the component together with built-in blueprints.

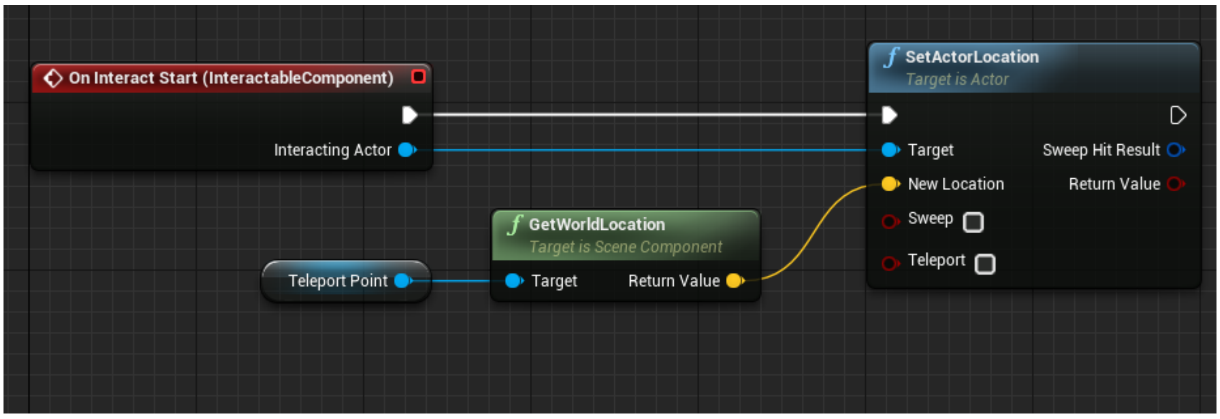


Figure 2: A very simple example that shows how the blueprint visual scripting looks like in UE4

A very similar system of node-based graphs is used for building complex materials in UE4. A material describes how an object, that uses it, looks like. It can for example define how it reacts upon light, whether the object appears to be rough or specular. The advantage of UE4's approach, using a node-based abstraction for materials, is that it makes life easier for developers when creating and authoring styles or effects. Without this system in place it would require a graphics programmer to write a shader. There are many different kinds of shaders but to keep it simple and because it is not necessary for now it is enough to say that a shader is a program that is executed on the GPU and defines which color a pixel shall display at the end. Writing such shaders is not a trivial task. The node-based approach puts a layer of abstraction upon this process that allows developers to work with materials without needing to know the low-level internals of shaders.

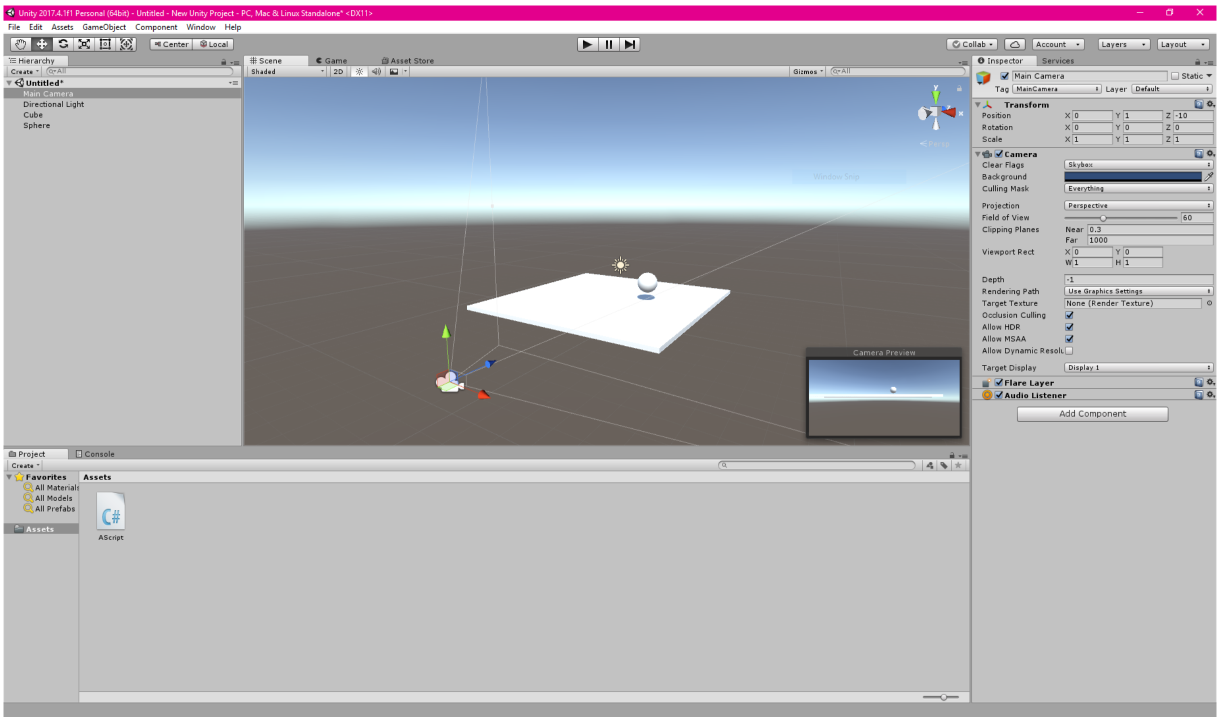
It should not be unmentioned that working with UE4, especially when needing very specific features or techniques that are not exposed to the abstraction systems, it requires a fond knowledge of C++ and the internal engine systems to fulfill the task. UE4 has grown in the past years and due to the fact that its source code is authored by many developers, both Epic engineers and open source contributors, bug reports are likely prioritized in relation to the needs Epic has for its own products developed in UE4. The documentation is well written for engine tools and visual scripting but is a little weak on the C++ side. When working directly in C++ sometimes compile-times can decrease iteration times in UE4 which is due to the Unreal Build Tool (UBT) and some default configurations on how to deal with precompiled headers and unity builds.

To conclude this paragraph about UE4 it can be said that Epic created an engine that is suitable for creating games from any genre, which sometimes require tweaking the engine's source code. UE4 offers a variety of tools and integrations for asset authoring software which makes it easy for developers to start working with it. The abstraction systems paired with the revenue share licensing model lowered the entrance barrier for new game developers and are a reason for why the engine is in such a good market position. The development experience and fast iteration times fade away the more direct C++ coding is involved but again a trade-off between developing every system in-house and dealing with problems that rise has to be made.

2.2.2 Unity

The biggest competitor on the market for UE4 is Unity, and there are several reasons why there is a second product that viable. Unity was born in 2004 and created by the company called *Over the Edge* which was later re-branded to *Unity Technologies*. Contrary to the evolution of UE4 that evolved from moddable games and with easy extension in mind, Unity was created to lower the access barriers for 2D and 3D game development. After their first game, GooBall, failed commercially the founders discovered how valuable engine software and tools are. This led to their decision of building an engine that is accessible to as many people as possible and that promises ease of development and cross-platform support. With these principles in mind Unity became a product that was adopted by many developers and nowadays is used by many studios and small developers.

In contrast to UE4 the source code of Unity is not freely accessible but it can be acquired by acquiring an enterprise subscription. If source code access is not needed, which is the case more a major percentage of the developer products, Unity offers a very fair licensing model. A personal license can be registered for free, with the constraint that a product does not generate more revenue than 100.000$ annually. This version includes all Unity core features as well as continuous upgrades and access to Unity beta versions. The next tier, called Unity Plus, includes everything of the previous one and adds more flexible customizations (custom splash screen, pro editor UI), more in depth analytics and it alters the cap of concurrent players on multiplayer games hosted by Unity. The Plus tier costs 35$ per month and is constrained by a revenue cap of 200.000$ per year. If the income exceeds that limit, the Unity Pro tier comes without any revenue limit at all for 125$ per month. It adds again more in-depth analytics and alters the concurrent players cap once more. Independent of the selected tier, Unity includes an editor to interact with its tools and the game world.



Main Camera Static
Tag MainCamera Layer Default

Transform
Position X 0 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags Skybox
Culling Mask Everything
Projection Perspective
Field of View 60
Clipping Planes Near 0.3 Far 1000
Viewport Rect X 0 Y 0 W 1 H 1
Depth -1
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling ✓
Allow HDR ✓
Allow MSAA ✓
Allow Dynamic Resolu
Target Display Display 1

Flare Layer
Audio Listener

Figure 3: The default layout of the editor shipped with Unity

The editor is quite similar to the one of UE4 but the views and tools differ in their naming. Going from left to right and top to bottom Figure 3 shows the Hierarchy, collection of game objects that are placed in the game world, the scene or current world, the Inspector, a list of components attached to a single game object and the project structure showing folders and assets. What was called an actor in UE4 can be compared to a game object in Unity. A game object is an entity that can be placed in the world but does not contain any logic itself. Rather it is a container that holds a collection of components, each holding data or logic. This system of entities and components is called an Entity Component System and will be described in detail at the end of this chapter. Because of the ECS components can be shared and reused among projects which simplifies the development process and cuts iteration times when done properly. It was already mentioned that components can also contain logic which already describes the basics on how game rules and logic are created in Unity. Where UE4 provides a visual scripting system Unity does not have anything similar. In Unity scripting is done in C#, a high-level programming language running in the MonoDevelop/.Net runtime. The work flow of creating gameplay logic often starts with a programmer creating a new scripting component in C# which then later can be used by designers to assemble new game objects without needing to touch any C# code. This is ensured by a system that allows programmers to expose certain properties of the script to the editor, where values can be tweaked by designers to fit the needs of the game. Although Unity does not have a visual scripting system it is easy to create and run scripts because they are contained withing a single component and an error inside of them will not crash the whole engine but is guarded by the scripting runtime.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AScript : MonoBehaviour {
    // Use this for initialization
    void Start () {
    // Execute code at component start ...
    }

    // Update is called once per frame
    void Update () {
    // Execute code once per frame ...
    }
}
```

Code 1: Example of an empty C# script in Unity

While previous versions of Unity supported two additional scripting languages, Boo and Javascript, C# was the most dominant scripting language used among Unity games.

A long time it was the node-based material graph that separated Unity and UE4, but since version 2018.1 (that is in beta-state at the time of writing) Unity introduces the *Shader Graph*, seen in Figure 4, which serves as a visual interface for building shaders. Together with the *Scriptable Render Pipeline* and the new job system, programmers gain more access over low-level and performance critical parts of the engine which closes the gap to UE4 a bit more regarding performance and control.

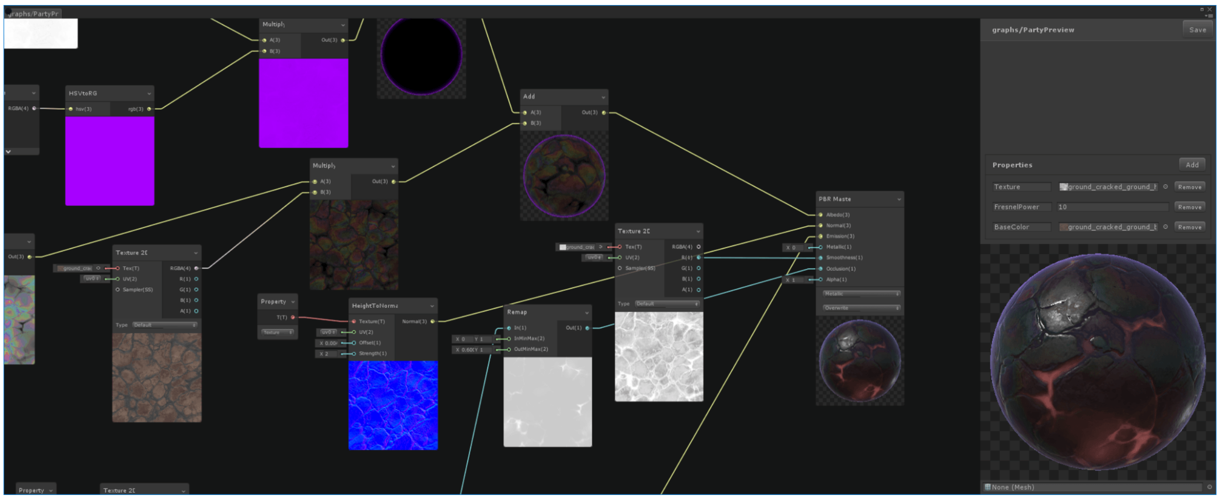


Figure 4: The new visual shader graph editor that ships with Unity 2018.1

Together with all these features and the asset store, a marketplace hosted by Unity where developers can upload, download and sell components, art and different plugins Unity is a tool suitable for games from any genre. Its low entrance barriers and support for many different platforms make it a powerful competitor to UE4. It is also due to these aspects that Unity is the tool of choice for game developers that plan to release on mobile platforms, such as Android or IOS. Unity is also the engine often chosen for rapid prototyping due to good iteration times and the asset store.

2.2.3 Molecular

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.2.4 Tombstone

The Tombstone engine is developed by *Terathon Software*, which was founded by Eric Lengyel in 2001. It is a cross-platform engine written in C++ that runs on Windows, Playstation 4, Linux and MacOS. This engine is mentioned in this chapter because it features an interesting architecture and invented and uses several innovative and robust techniques. The Tombstones engine architecture is separated into several layers of managers combined with general utility libraries and a plugin system. General utilities such as memory management and a container library build the bases for managers of different systems. These system manager include for example the resource, thread and graphics manager. Upon this layer more high level managers are built that handle the game world, the scene hierarchy or the animation system. To work with these systems Tombstone offers several plugins, shipped with the engine.

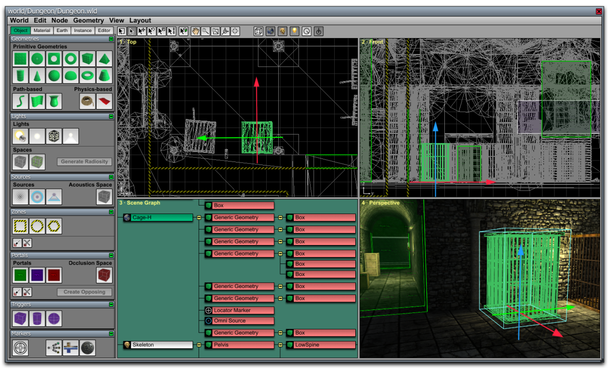


Figure 5: The world editor plugin used to edit worlds in the Tombstone engine

Among these there is also the world editor that, similar to the bigger engines, allows for creating worlds and levels. Beside the world editor other plugins are a visual shader editor, several tools for importing assets and UI builder to generate panels and widgets. Beside the well-defined architecture Tombstone features two standards also created by Eric Lengyel, named the Open Data Description Language (OpenDDL) and the Open Game Engine Exchange Format (OpenGEX). These protocols are used for exchanging data with asset authoring software like Maya or Photoshop and to store serialized data. The design of the architecture if very well described in a visual form directly at the Tombstone homepage (http://tombstoneengine.com/architecture.php).

Contrary to the bigger companies the Tombstone engine's licensing model does not offer free versions. To acquire a lifetime license for all platforms (Windows, Linux & MacOS) a fee of 495$ per person has to be paid. This license does not include any revenue share nor is it limited to a specific amount of shipped games.

The Tombstone engine is an example of a quality piece of software and showcases in what direction an engine can develop if the design goals are carefully thought out. It also tries to establish open standards for exchanging data between different parts of the game development toolchain. If such standards would be implemented and obeyed by more engines it would ease the process of migrating the workflow of engineering teams to another engine or software that would better fit their current project.

2.3 Rust game engines

Because the goal of this thesis is to research whether Rust is a potential choice for a game engine's main language the author wants to highlight work already done in that field. All previously described solutions are driven by C++. While some of them use it for the entire engine, others only built their core systems in C++ and use higher level languages on top of it. To showcase what was already done in the Rust ecosystem and to better understand some decisions the author made when implementing the submodules this section is dedicated to the two biggest engines written in Rust - Piston and Amethyst.

2.3.1 Piston

2.3.2 Amethyst

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.4 Tools and asset pipeline

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.4.1 Editor

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.4.2 Asset pipeline

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.5 Engine subsystems overview

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.5.1 Memory Management

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Custom Allocators

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.5.2 Job System

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.5.3 Rendering

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.5.4 Entity Component System

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.5.5 Scripting

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3 Rust

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.1 Current state

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.2 Rust ecosystem

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.3 Concepts

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.3.1 Borrow checker

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.3.2 Traits

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.3.3 Hygienic macros

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.4 Pitfalls

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4 Subsystem implementation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.1 Spark engine architecture

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.2 Development environment

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.3 Rendering framework

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.4 Memory Management

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.4.1 API

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.4.2 C++ implementation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.4.3 Rust implementation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.5 Containers

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.5.1 API

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.5.2 C++ implementation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.5.3 Rust implementation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.6 Entity Component System

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.6.1 API

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.6.2 C++ implementation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.6.3 Rust implementation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5 Measures & Comparisons

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.1 Environment

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.2 Testcases

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font,

how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.3 C++ performance

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.3.1 Memory Management

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.3.2 Container

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.3.3 Entity Component System

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font,

how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.4 Rust performance

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.4.1 Memory Management

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.4.2 Container

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.4.3 Entity Component System

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font,

how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.5 Comparison

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6 Conclusion

Bibliography

[1] J. Gregory, *Game Engine Architecture*, 2nd ed. 6000 Broken Sound Parkway NW: CRC Press, 2014.

[2] B. Stroustrup, *The C++ Programming Language*, 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2013.

[3] O. Blandy, Jim, *Programming Rust - Fast, Safe Systems Development*, 1st ed. OReilley, 2017.

[4] Chapter-wise on usage, *Game Engine Gems 3*, 4th ed. 6000 Broken Sound Parkway NW: CRC Press, 2016.

[5] D. Portisch, “Multitasking using a job sytem with fibers,” Master’s thesis, Fachhochschule Technikum Wien, Höchstädtplatz 5, 1200 Wien, 2017.

List of Figures

Figure 1 The editor shipped with UE4 4
Figure 2 A very simple example that shows how the blueprint visual scripting looks like in UE4 5
Figure 3 The default layout of the editor shipped with Unity 7
Figure 4 The new visual shader graph editor that ships with Unity 2018.1 8
Figure 5 The world editor plugin used to edit worlds in the Tombstone engine 9

List of Tables

List of Code

Code 1 Example of an empty C# script in Unity 8

List of Abbreviations

ECS Entity Component System

GB Gigabyte

RAM Random-Access-Memory

GPU graphics processing unit

FPS first person shooter

RTS real-time strategy

UE4 Unreal Engine 4

UI user interface

UBT Unreal Build Tool

UHT Unreal Header Tool

OpenGEX Open Game Engine Exchange Format

OpenDDL Open Data Description Language