Tools & Technology 3 – Performance Requirement

Kevin Hagen, WiSe 2021

Introduction to Game Programming

Carefully read through the whole assignment. If you have any questions, post them in the teams channel or contact me directly.

Submission Facts

Deadline: 18.02.2022, 11:55 p.m.

Form: Written report and a link to your git repository

Where: Post your submission in the channel *Homework and Assignments* with the following convention:

FirstName_LastName_PerformanceRequirement – Repo: LINK

Attach your report to the message and name the document like so:

FirstName_LastName_Report_CA

Report: The report should be 5 (+/- 1) pages long. The first course presentation contains a detailed and comprehensive list of what and how I expect your report. Submit your report as PDF.

Grading Criteria: As explained in the first lesson. If in doubt, you can check the presentation.

Problems & Questions: Can be discussed either openly in the Homework Discussions Teams channel or directly with me.

One submission per student!

Task

This semester we delved deeper into the realm of game development and had our first look at programming languages. We had a closer look at **C#** and **Object Oriented Programming (OOP).** These are two core concepts that should be well-known to every (Unity) Game-Programmer. This performance requirement aims to deepen your understanding of these topics, while giving you the freedom to add complexity on your own, in such a way that it best suits your level of skill.

Your task will be to implement a feature set from a small game as outlined in this document. The outline should have sufficient level of detail, if you are missing some, feel free to reach out.

The report:

- Structural details of the report can be found in the presentation from the first lesson.
- In terms of content, it should contain…
    - … an analysis of the required feature set, its components and how you are planning to realize the concept of the game in code
    - … details on ***your*** **implementation** of the feature set.
    - … things you might have changed.
    - … a conclusion of your work.

General Info:

- You may choose to do the assignment in Unity Bolt or by writing C# code
- You ***may*** use third party assets (e.g. DOTween) for parts of your implementation
- They ***may not*** make up the majority of your code
- You ***may*** use any non-code related third party assets (audio, animations, VFX, …)
- Record & upload ***video footage*** of your ***working*** project implementation
- If something does not work post hand-in, you get ***one*** chance to fix it. If this is the case, you will be contacted with more details on this.
- Commits and already w.i.p. ***before 23.12.2021 08:00 p.m.*** won't be considered!
- If you are a more experienced student, feel free to try and push your boundaries by extending the scope of the project. Since there are many different skill levels present in our class, I do not offer "one specific" addition but I'd rather want you to have a look at what *you* think is challenging you and your current skill level. Examples I can think of are procedurally generating the level (or parts of it), adding more complex enemy types/behaviour (FSM, BTs, GOAP, …), adding more depth to the character controller, …
-

Best Practices

DOs

- Hand-In your assignment in time.
- Stick to the proper form of documentation (Cover Page, ToC, …), as specified in the first lesson.
- Use git, commit often and hand-in the whole repository.

DONTs

- Hand-In a half-baked documentation (horrible formalities, incomplete sentences, “denglish”, obviously not proof-read, …)
- Forget the .gitignore or set it up incorrectly (see first lecture on how to do so, if you forgot that)
- Upload a file archive (.zip/.rar) instead of actual VCS usage
- Leave your code uncommented
- Hand-In compile errors (broken/not working is fine, but NOT compile errors!)

Violating any of the above stated DONTs results in an immediate failure of the assignment (5.0). This also means, NOT sticking to the DOs will also result in immediate failure.

DISCLAIMER: Do not worry, if you make *some* mistakes (e.g. spelling) they will of course be tolerated. This is mainly to prevent large and unnecessary errors due to carelessness, as they happened to a large extent in the previous semester.

Project Outline

Create a new Unity Project named „TT3_Performance_Requirement“

The game is a 2D side scrolling platformer in the spirit of the old mario games. There are the following main entities in the game:

- PlayerCharacter
- Enemy A
- Enemy B
- Level Goal
- Coins
- Player Upgrade A
- Player Upgrade B
- Regular Platforms
- One-Way Platforms

The Player Character

- Can walk left/right
- Holding down shift makes the character sprint
- Pressing spacebar makes the character jump
- When hit by an enemy, the player dies, restarting the level from the start

Enemy A

- Constantly walks to the left, playing a walking animation
- Has an adjustable speed
- Jumping on top of his head kills him, playing a kill animation and then removing the enemy from the level

Enemy B

- This enemy patrols left to right in a given zone.
- It should either turn left/right when hitting an obstacle in the level or when it has reached its patrol point
- Tip: You can use transforms to restrict the area of its movement
- This enemy can not be killed by jumping on the player, instead the enemy would then damage the player
- It must be killed by utilizing Player Upgrade A

Level Goal

- When reaching this level goal, the next level should start
- If we are finishing the last level, a short “credit scene” should be shown. Simply list your name, and some assets if you used any
- After 10 seconds the credits scene should close and load the main menu again

Coins

- Coins represent player score. They should be scattered throughout the level
- Coins can be collected by walking through them

- If you collect a coin, a UI element on screen should be updated and show the current amount of coins/score the player has
- there should be 3 types of coins:
    - Regular - awards 1 coin
    - Medium - awards 5 coins
    - Large - awards 10 coins
    - they should all have their own graphic

Player Upgrade A

- When collected, the player grows and gains an extra life - this is consumed before dying. Once hit, the player also looses the upgrade and shrinks to regular size again
- If the player has the upgrade, it should also be shown in the UI somewhere (show a small sprite that represents this upgrade)
- Having the upgrade enables the player to shoot bouncing projectiles, similar to marios fire balls
- The skill can be used by pressing the e button

Player Upgrade B

- When collected, the player grows and gains an extra life - this is consumed before dying. Once hit, the player also looses the upgrade and shrinks to regular size again
- If the player has the upgrade, it should also be shown in the UI somewhere (show a small sprite that represents this upgrade)
- Having the upgrade enables the player to perform a stomp attack while in the air
    - The stomp attack rapidly moves the player to the ground
    - If you can, implement a small screen shake, particles, sfx, … to support the momentum of this attack
    - It acts like a “shock wave attack”, everything in close proximity to the player should be killed/destroyed. direct hits also kill/destroy the element
- The skill can be used by pressing the e button while mid-air

Regular Platforms

- Used for regular platforming
- The player can stand atop of them
- They do not move

One-Way Platforms

- Platforms with collision from only one side
- You can jump on them from beneath
- Pressing the down key makes you fall through them

Other

- There should be a killzone beneath the level to kill the player or other entities when they fall out of the map
- There should be a total of 3 different levels to play through
- The main menu should have a UI with a start button and a quit button. Start loads the first level, quit exits the application
- The result should be a playable build
- Building the levels is up to you. Playtime should be something between 5-10 minutes

- Implement SFX/music for the game and its elements where you feel its suitable - anything that needs communication with the player, like collecting coins, killing enemies, using things, … is a good place to start.