Lab 3 - Game Project in 2D

Endless Runner

Student Name(s)

Enter your name(s) here (All team members):

Name	Email	GitHub User Name
(insert name)	(insert email)	(insert github username)

Instructions for Teacher

(insert, if needed, instructions for teacher here)

Introduction

The goal of this assignment is to write anendless runner game in 2D, using C++ and SDL, with an emphasis on **game feel**. An endless runner is a specialization of the platformer genre where the player is always moving forwards. The genre was popularized the game Temple Run.

The reason for picking this genre is that the core mechanic is easy to understand and easy to implement - we can have a simple input scheme with just one button - this lets us focus on the feel of the game. We try to maximize how the game is perceived using various audio-visual effects and techniques such as screen shake and fitting sound effects.

Go check out wikipedia or itch.io for inspiration.

Reading

- SDL2
- SDL2 Tutorial
- git - the simple guide
- Markdown Cheat Sheet
- Sprite Sheet Wikipedia
- Endless Runner
- Game Programming Patterns

Pass

The following features are required for a**pass**

Win/Lose Conditions

The player must be able to win in some way and in the case of a game like a endless runner - a high score of some kind - length run or time survived. In this type of game the lose condition then becomes not being able to continue "running".

Main Game States

There must be at least three different**main** game states in the game e.g*MenuState*, *PlayState* and *GameOverState*. These states can, and probably should, have sub-states such as the*PlayState* having a *PlayPausedState* and the *MenuState* having a *MenuSettingsState* and *MenuReplayState*.

Game States

Menu Easing

Easing is usually added to user interface elements and menu items to make the presentation look more interesting. Easing being a term coined by Robert Penner that is referring to different functions that adds flavour to motion.

Easing Functions in C++ Explanation of Easing Functions

Collision Detection and Resolve

A game is not complete without good collision detection and resolve in 2D. This is achieved by using algorithms to find if two colliders intersect or overlap, and then resolve the potential intersection by moving either object, depending if one of them is a static non-movable object or not, the penetration distance so that they no longer intersect. The collision detection algorithms developed in previous assignment comes in handy.

Keyframe Animations

The game needs to have at least three different animations with three or more keyframes. The behaviour for the animations can be the same, i.e they can all be looping. The animation code from previous assignment comes in handy.

Background Parallax Scrolling

The background in the play state must be usingparallax scrolling with three or more layers.

Screen Shake

In the games play state there should be some kind of screen shake effect during game play. Remember that some people are sensitive to flashing lights, so don't over do it.

Configuration File

The game must read a configuration file that has at least three settings required for the game. E.g size of the window as width and height (note: this is one setting since it does not make sense to have one without the other) or some factor of the amount of screen shake allowed et cetera.

Music and Sound Effects

Background music must be played and appropriate sound effects must be played when some kind of event happens in game, e.g player shoots or an enemy is hit by bullet.

Pass with Distinction

All features of **Pass** and the following features are required for a**Pass with Distinction**

Particle Effects

Two or more particle effects should be rendered in the game. E.g shattered glass when the player jumps though a window or the smoke in the background in Canabalt. But it can also be something in the main menu or even in a loading screen.

Replay Recording and Playback

During play the input of the player is recorded first to a buffer in memory then, when the player hits one lose condition, the content of the recording (i.e memory buffer) is written to file on disk.

This newly recorded and saved replay, or a previously recorded replay, can then be selected in the main menu and played back. The replay feature should have a history of replays to select from.

TL;DR for the features required for different grades

Pass

- Win/lose conditions
- 3+ game states
- Easing
- Collision detection and resolve
- 3+ different keyframe animations
- 3+ layers of background parallax scrolling
- Camera shake
- 3+ settings configuration file
- Music and Sound Effects

Pass with Distinction

- All features of Pass
- 2+ particle effects
- Replay recording and playback

Instructions

Get started

Every student has to do the following steps:

1. Make sure you have a GitHub account.
2. Accept the assignment by following the following link:https://classroom.github.com/g/P09AS6S4.
3. Select yourself in the list of identifiers. *Make sure you select the correct one*
4. Create a new Team *OR* Select your team from the list. *Make sure you select the correct one*
5. If this is your first time. You will receive an email with an invitation to the organisation `gameprogrammingii`. Accept this invitation.

The assignment

1. Clone the repository
2. Edit the README.md (this file) and enter the name, email and username of all students in your team.
3. Complete the assignment (read the rest of this document).

Hand-in

1. Commit and Push all your changes
2. Create a new Issue in your GitHub project indicating that you are done with assignment and tag JohanNorberg

Deadline

2020-03-27

Grading

This assignment has Pass, Pass with Distinction or Fail. For a Pass, all the required features in the Pass section of this document must be implemented correctly. For a Pass with Distinction, all the features for Pass must be implemented correctly in addition to all required features for Pass with Distinction. Failing the steps defined in the Requirements section and Hand-in section will lead to a Fail.

Requirements

This is a individual or group assignment. Original code written by the student(s) with C/C++. Example code and provided external code by course responsible can and should be used as a starting point for the laboratory.

Plagiarism

At university, we are continuously engaged with other people's ideas - we read about them, we discuss them in class and we write about them in our assignments. It is therefore important that we acknowledge these people in our assignments/projects/papers that we submit for marks. If we do not adhere to these basic requirements, we are making ourselves guilty of a gross violation of the academic standard. Academic dishonesty in any form including, but not limited to plagiarism and collusion, cheating in tests, examinations and assignments, theses and research papers, is regarded as a serious offence and will be dealt with in terms

of the provisions of the University’s Disciplinary Rules for Students.