**Champlain College - Lennoxville**

**Assignment 5: Boulder dash**

|  |  |  |  |
| --- | --- | --- | --- |
| **PROGRAM:** | 420.B0 Computer Science Technology | | |
| **COURSE:** | Game Programming 1 | | |
| **COURSE CODE:** | 420-440-LE | | |
| **WEIGHT:** | 7% of the final score | | |
| **SEMESTER:** | Winter 2025 | | |
| **INSTRUCTOR:** | Francis Gauthier  fgauthier@crcmail.net | Office C-308 |  |

# Objectives:

* Practice using Git source control to work on a game project
* Practice working in teams and splitting tasks amongst members
* Practice many course concepts and scripting logic seen throughout the semester
* Practice making more complex games, with game menus, mechanics, high scores and exported

# Goal

The goal of this assignment is to complement a simple mini game where a character can be moved and must survive in a small space where rocks are falling from the ceiling. The goal is to complexify this game in many aspects learned in the scope of the course.

The goal is also to practice working on game development in teams, with source control. The scope of this assignment is large, time-consuming and meant to be achieved in teams of 2 students.

# Project setup

For this assignment, use the project template provided. It contains a base game and many assets that can be used. You will also need to find some assets for some tasks listed.

## Steps to follow

1. Read the list of requirements for the game, listed below
2. Identify the list of individual tasks required to complete the assignment
   1. Assign each task to a specific student of the team
3. Submit your list of tasks in LEA
4. Create a private GitHub repository
   1. Add your 2 team members as collaborators
   2. Add ***frangauthier***as a collaborator
5. Implement the different tasks simultaneously in the project using source control
6. Actively push your final result in GitHub until the submission date

# Game requirements

## Implementing losing conditions

The character can lose the game in two ways:

1. The character reaches the top of the screen
   1. When the character would go off the screen viewport, then they lose right away.
2. The character was damaged by too many rocks
   1. The character must be damaged by falling rocks.
   2. Each rock should deal one damage to the player. The player can receive 4 hits before the game is interrupted (game over).

When the game is lost:

* Switch to a *Game over* scene, which contains:
  + Display a label saying “Game Over”
  + Display a button that allows you to restart and make another attempt at the game
  + Display a button that allows you to return to the main menu
  + The UI should be visually centered

## High Score mechanic

When the character goes deeper in the play area, the depth of the player should be recorded.

For each layer of rock traveled down, a depth meter should be updated to display the deepest level reached, in meters. For reference, try to provide a realistic number of meters. For example, one layer of rock should be ~3 meters high.

Provide a simple UI label that will display in real-time the deepest level reached:  
Example:

|  |
| --- |
| **Current depth: 20m** |

When the character loses, the *Game over* scene should display:

* The current score
* The highest score reached previously, amongst any game session on the same PC (local save data)
* The high score should also be updated in the main menu.

## A UI health bar

Add to your screen overlay a health bar displaying the current health of the character.

The health bar should be animated, i.e. no sudden jump from one value to the next.

The health bar should be always visible and be accurate.

Examples of desired result:

A red line on a gray background

Description automatically generated 

## Health potion pickup

To provide incentives to dig certain areas of the game, implement a health pickup that can restore some life.

* A red liquid in a bottle

  AI-generated content may be incorrect.Create a health potion pickup scene
  + Use an Area2D
  + Set your layers and masks properly
  + Restore completely the health value of the character on contact with the pickup
* The health pickup can be spawned in the location of a rock tile. Locate the script that spawns the rock tile. There is a logic that leaves an empty gap 20% of the time. Use that logic to spawn a health potion randomly instead of an empty tile with a percentage of chance. For example, 10% of the empty tiles are replaced by health potions spawns.

## Hurt and invincibility animation

To help the player, implement this invincibility mechanism:

When the player is hit by a rock:

* A short sound effect is played (use *negative-guitar-tone* in Assets/Audio)
* The player cannot be damaged for the next 3 seconds
* The player sprite is animated to let the player know that they are invincible. The animation is synced with the invincibility.
* The health bar is updated accordingly
* Example of animation:



## User interface, fonts and localization

In the main menu, you must add two buttons: **EN** and **FR** to switch the game language. The language does not have to be kept as a setting between game sessions. The game should be in English by default.

Every text used in the game must be localized and translated.

The game title uses a specific font. You must apply the same font to every text label and button label in the game.

## A black screen with white text AI-generated content may be incorrect.Visual effects

You must implement 2 specific visual effects in the game:

1. A scene transition animation. Using the **SceneManager** plugin, you must implement animated scene transitions when switching scene, which includes:
   1. From the main menu to the main game
   2. From the main game to the game over scene
   3. From the game over to the main game (restart)
   4. A close up of a finger pointing

      AI-generated content may be incorrect.From the game over scene to the main menu
2. A **particle system** that emits small rock fragments when the falling rock disappears. Instead of simply disappearing, the rocks who hit the player or the rock tiles should create a small explosion of small pebbles when disappearing.

## Game export

Before submitting, the game must be built into one of the following platform: Windows or Web.

If the game is exported to Windows, the build files must be included in the file submission in an obvious location. If built for web, the game must be hosted in Itch.io, made *public* or *restricted (with password)* and the URL link must be submitted with the project (*with password if restricted*).

# Balancing the game - optional

The original game has been quickly balanced. You can tweak the game to make it easier or more difficult. Things that you can adapt freely:

* The spawn speed of the falling rocks
* The character speed and jump force
* The speed of the camera (scrolling vertically)
* The proportion of rock versus empty space versus health potions

# Splitting work

Before heading onto the implementation of the requirements, you must decide on a teammate for the assignment. Next, you must break down the requirements into subtasks that can be assigned to one of the teammates.   
1 - Fill the table below before your implementation. Add a task description and an assignee. Aim for a fair split of the tasks workload between team members.

2- Prioritize the tasks. The tasks at the top of the table represent high-priority task that should be executed early on. Tasks at the end of the table should be done last.

When filled, you must submit this file in LEA (one per team) before implementing the requirements in code.

|  |  |  |  |
| --- | --- | --- | --- |
| Team members | Simon Ares | |  |
| **Description of the task** | | **Assigned to:** | |
| Ex: Creation of the GitHub repository | | Simon Ares | |
| Everything | | Simon Ares | |
|  | |  | |
|  | |  | |
|  | |  | |

**\*Add as many rows as needed.**

# Evaluation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Performance criteria** | **Poor\*** | **Basic** | **Great** | **Perfect** |
| **Analyze the application project**  Accurate identification of the tasks to complete from the analysis of document  Fairness of distribution of tasks  Accurate prioritization of work | 6 | 12 | 16 | 20 |
| **Prepare the computer environment development**  Proper setup of the Git repository  Proper import of external source code (plugins)  Proper import of external assets  Proper creation of a game build | 4 | 8 | 12 | 15 |
| **Generate real or virtual world representations**  Appropriate use of game menus, scene transitions  Appropriate use of control nodes and UI elements  Appropriate use of fonts and themes | 4 | 8 | 12 | 15 |
| **Program the game or simulation logic**  Number of requirements achieved correctly  Proper programming of visual effects and animation techniques  Proper programming of sound effects  Proper application of localization techniques  Proper use of collisions, signals and scripting logic | 15 | 30 | 40 | 50 |

# Submissions

The breakdown of tasks must be submitted before **Tuesday April 8th, End of day**, in LEA.

Code submission must be made in GitHub. Make sure to add *frangauthier* as a collaborator to your repository.

Code submission must be made before **Wednesday April 16th, End of Day**. The projects will be cloned when corrected. Any commit made past the due date will be considered as a late submission.

Late submissions are accepted with 20% penalty if no agreement was reached with the teacher.