Champlain College - Lennoxville
Assignment 2: 3D game

PROGRAM:	420.B0 Computer Science Technology	
COURSE:	Game Programming 2	
COURSE CODE:	420-540-LE	
WEIGHT:	12% of the final score	
SEMESTER:	Fall 2023	
INSTRUCTOR:	Francis Gauthier fgauthier@crcmail.net	Office C-239

Objectives:

- Practice building a simple 3D game in Godot
- Practice using 3D meshes, materials and texture maps
- Practice working with the 3 axis of 3D games
- Practice using animations, particles systems and other special effects
- Practice working in teams and sharing code

Tasks:

1. Program the character movements and animation
2. Program two objects that our character can interact with and complete objectives
3. Design and build the world where our character evolves
4. Assemble and build your game, following industry standards

A more detailed explanation of the task will follow.

Teams

This assignment will be made in teams of 4. The project has been split in 4 tasks that are equally important for the result and each task will be worth 25% of the final grade of the assignment. These four tasks were split to be able to make them separately without too much dependency on other tasks (except the last one).

Important - Licenses

Most of the work for this assignment can from external meshes, materials, textures, animations, etc. available online. Each asset used must be credited to the maker.

At the root of the project, in the Readme.md, for each asset used, you must specify:

- The author's name(s)
- A link to the asset (where you took it)

Failure to do so may result in a Plagiarism and Cheating case and a grade of 0 for the whole team.

The tasks

Character control & animations (25%)

A 3D character that can move and look around smoothly with animations

Visual requirements:

- The character is composed of a 3D mesh that represents a complex character (humanoid or animal-like)
- The character scene is either a RigidBody3D or CharacterBody3D
- The character has a collision shape that follows roughly the character mesh without being too complex (too many polygons)

Control requirements:

- The character can be controlled using the WASD keys or the arrow keys.
- The character can go in most areas of the map, in the X-Z plane.
- (Optional) The character can jump to access higher locations.
- The camera can also be controlled with other keys or the mouse to look around the character.

Animation requirements:

- The character has at least 4 different animations
- The character animations are triggered automatically or upon pressing a specific key
    - For example, the run animation is automatic when the character velocity is high
    - For example, a punch animation is triggered when pressing a specific key
- All animations can be seen in-game

Specifications:

- The character mesh and animations can be downloaded and imported from an online library or asset store, following the license rules of the assets. No paid resources are allowed.

Interacting with objects in the scene (25%)

Two objects that make the world interactive and that add depth into our world

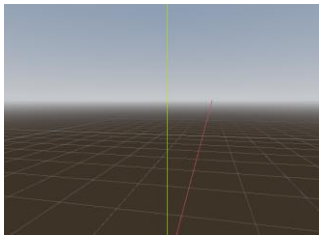
Your task will be to create incentive to play the game through objects that can be interacted with in game. You will also add a simple UI to keep track of the objectives.

The objects requirements (for each object):

- Are built in separate scenes
- Each object can be interacted by moving into a specific area or by pressing a specific action button in that area
- Each object should trigger a minimum of two special effects to enhance the game. Special effects can be:
    - Sound clips played
    - Tweens
    - Animations
    - Particles emitted
    - Lights
    - Physic-based movements (pieces falling, colliding, etc.)

Examples of object interactions:

- Collecting/harvesting some fruit
- Destroying an object to access an area
- Helping a plant to grow
- Catching a moving enemy (ex: rat)



When completed, the objects should be easy to replicate (CTRL+D) to add many instances in the game in different locations.

Simple In-game UI

To help the player to keep track of the objectives, add a simple in-game UI.
UI requirements:

- Overlay (always visible)
- Gives indications on what to do
- Tracks progress (through a progress bar, counter, icons, etc.) on the objectives

No need to do anything when the objectives are achieved. The progress is simply to give incentives for the player to explore the scene.

World design and environment (25%)

An immersive experience that creates the atmosphere of the game

World requirements:

- The world has boundaries. The main character cannot leave the boundaries.
- The world is large enough to be able to differentiate sections of the map.
- The world uses a mix of at least 3 different materials (ex: rock, grassy, dirt)
- The world contains at least one of each:
    - Static objects that cannot be moved (ex. trees)
    - Dynamic objects that can be pushed around (ex: wooden crate)

Environment requirements:

- The world contains a directional light that cast shadows
- The wolrd contains a panorama (HDR) for the sky
- The world contains background music that matches the world design
- (Optional) Add one environmental hazard to your world like fog or rain/snow effects over a small period
- (Optional) The sky color may change over time to simulate dusk, dawn, or day-night cycle

Assemble and build your game (25%)

The less flashy, but quite important part of a team

Your task will be to first build a game menu, then collect the scenes of your teammates and assemble them in the final version of your game. Finally, make a build of your game when completed.

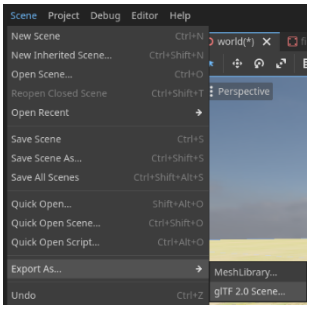
Main menu requirements:

- Ability to start the game
- Ability to set the volume of the game
- Ability to quit the game
- Menu theme fits the atmosphere/genre of the game

Assembling the game

You can either:

1. Use git as version control. (recommended)
    a. Create a git repository
    b. Add your team members
    c. Help them with their tasks
    d. When the 3 first tasks are done and pushed, pull their changes, and assemble the main scene.



2. OR work with exported scenes.
    a. No need for git. You will create the main project.
    b. Help your teammates with their tasks
    c. Help them to export their scenes as GLTF files
    d. Import the scenes in your game (textures might be lost)

Building the game

Finally, complete your game by making a Windows build of your game.
The output should be a minimum of two files:

- A .exe executable file of your game
- A .pck file requires for the .exe to work

Output your build in a Build/ folder at the root of your project.

Submission

The file submission must be made through a GitHub repository.
Add frangauthier as a collaborator to your project.

Code must be pushed by October 15th 2023, End Of Day. Late submission accepted, 10% per late day.