The University of Saskatchewan

Saskatoon, Canada

Department of Computer Science

CMPT 306– Game Mechanics

Assignment Group 1

Game Clone

Date Due: October 31, 2025, 11:59pm

Total Marks: 6

Changelog

Oct 1, 2025 : Released to students
Oct 2, 2025 : Fixed typo in Harvest 306 specification where functionality for Hoe and Shovel were swapped.

General Instructions

- Assignments must be submitted using Canvas.
- Unless otherwise specified, programs should be written in GDScript and be compatible with Godot 4.4 or newer.
- Unless a specific framework or library is requested, all submissions should be compatible with the basic vanilla version of the engine.
- In some assignments, certain features in Godot will be limited or forbidden, usually because they circumvent the learning objectives of the assignment. The assignment will specifically state this where applicable, but ensure you are following the assignment instructions carefully.
- VERY IMPORTANT: Canvas is very fragile when it comes to submitting multiple files. We insist that you package all of the files for all questions for the entire assignment into a ZIP archive file. This can be done with a feature built into the Windows explorer (Windows), or with the zip terminal command (LINUX and Mac). We cannot accept any other archive formats. This means no tar, no gzip, no 7zip, no RAR. Non-zip archives will not be graded. We will not grade assignments if these submission instructions are not followed.
- Unless the assignment specifically asks for the entire Godot project, please only submit the files requested. The markers will have their own marking environment where they will drop your files into, if they have to navigate through your project structure they will deduct marks.
- It is expected that all assignment work is under git version control. Each assignment will have different expectations and will list them explicitely. A portion of each assignment marks will be a submitted git log that demonstrates that you followed the required development process.

Question 1 (5 points):

The purpose of this assignment is to practice using git in a small group, demonstrate proper development workflows, and track individual contributions of group members.

Your Task

One member of your group should create a git repository (host location is up to you, but git.cs.usask.ca is preferred). Each member of your group should be given access to the repository and clone it to their own machine(s).

There are no specific git expectations for this assignment, but a git log **MUST be provided in order to receive any marks for the assignment**. It is critical to have a log that demonstrates what each group member contributed to the project in the event of group conflict later in the term.

It is recommended the group spend time early in the project getting together to test workflows; ensure everyone is able to clone, commit, push, pull the repository; experiement with merging and resolving conflicts; and setting up the basic structure of the project. This step is very important to ensure a cohesive group dynamic and that every group member understands the expectations and workflow.

Evaluation

5 marks : Git log is submitted in its entirety, demonstrating the development progress for the entire milestone.

Provided Files

The instructor has provided no files for this question.

Files to Hand In

Include in your final submission a file named `gitlog.txt` containing all commits pertaining to this phase of the group project.

Question 2 (1 point):

The purpose of this project is to learn the basics of Godot by recreating a game reference, and follow design specifications.

Your Task

As a group, you will create a game in Godot, using the provided reference video, and specifications listed below. The goal is to be as precise as possible, to learn how to implement common game mechanics in Godot. Everyone in the group is expected to contribute approximately equal, and everyone should be involved in all aspects of the project (e.g., a group member acting as a dedicated artist but not writing any code is not acceptable).

As a group you are to choose between 2 games (references available on Canvas):

306 Survivor : a game resembling the basic mechanics of Vampire Survivors, a game in the *Bullet Heaven* sub-genre of games.

Harvest 306 : a grid-based crop planting simulator reminiscent of farming sim games such as Harvest Moon and Stardew Valley.

Each of the games is broken down in the following pages, with specifications and requirements for each.

General Expectations

Each game is unique in terms of its formal and dramatic elements, but there are some consistent expectations that both games need to satisfy:

Main Menu : When the game starts, the player should be encountered by a user interface including:

1. The title of the Game
2. A button to start the game
3. A button to open the *options* menu (more on this later)
4. A button to quit the game

Pause Screen : A user interface that gives players the following options. Pausing the game should stop all gameplay and disable all user input/interactions as long as the menu is open:

1. Resume/Unpause
2. Open the options menu
3. Return to main menu
4. Quit game

Options Menu : A user interface that is available from both the Main Menu and the Pause screen. The Options menu should support 2 options:

1. **Music Volume**: a slider that adjusts the global volume for all music
2. **SFX Volume**: a slider that adjusts the global volume for all sound effects

Using audio is probably unfamiliar to most students. You should research the concept of `Audio Buses` in the Godot documentation to understand how to play all audio through the same bus, and allow the settings to adjust the volume of the entire bus. Also, be aware that audio uses Decibels which is a logarithmic scale. You should look up a formula to translate from a linear scale (i.e., a UI Slider) to an appropriate volume level.

306 Survivors Expectations

The following sections describe specifications for the 306 Survivors game clone. This project is a more complete game, in terms of mechanics, but is less technical than the Harvest 306 game and individual mechanics are less complex.

Art

Player : You are free to choose any sprite for the player (the alien UFO from the preview video is provided)
Enemy : You are free to choose any sprite for enemies (the alient UFO from the preview video is provided)
Bullet : You are free to choose any sprite for bullets (the laser ring from the preview video is provided)
XP Pickup : You should create or *legally acquire*(with license text file) a sprite for XP pickups
UI : All UI should be made using the provided UI spritesheet

Audio

The following are requirements for Music/SFX. A link to a collection of game assets is included in the starter assets, but you are free to find your own audio, if you prefer (again, either create yourself or legally acquire):

- Clicking on a UI button (i.e., Main Menu, Pause Screen, Level Up)
- Opening/closing a menu (i.e., Options menu, Pause menu)
- Background music should be playing on loop in the Main Menu and in the game
- A sound should play when the player shoots a projectile
- A sound should play when a projectile collides with the enemy and the enemy takes damage
- A sound should play when the player takes damage
- A sound should play when an XP pickup is collected
- A sound should play when the player is moving, and stop when the player is not moving

Mechanics/Rules

The follow describe the basic mechanics that must be implemented to receive full marks:

Player :

- The player should move using WASD/Arrow Keys
- The player should rotate/tilt slightly to the left/right when moving in that direction (see posted video for a visual reference)
- The player should periodically check for enemies within firing range and if one exists shoot the projectile towards the closest one. If no enemies exist, the player should shoot in a random direction.
- The player should have health, receive damage from enemies, display a game over menu if health reaches 0

Enemy :

- The enemy should move towards the player
- If the enemy gets within a specified distance of the player, they should stop and begin an attack routine, periodically dealing damage as long as the player is within the enemy's attack range

- The enemy should have health, take damage whenever a projectile hits them, and despawn when the health reaches 0

XP Drop :

- When an enemy is defeated, there is a random chance of spawning an XP pickup where the enemy was despawned

Enemy Spawning :

- The enemy spawner should have it's own regular heartbeat (e.g., every second)
- The spawner should have a capacity, the maximum number of enemies that can be in play at one time
- The spawner should maintain an Array of all enemies (add to the array when they spawn, remove before an enemy despawns)
- Each heartbeat the spawner should check if there is room for another enemy (compared to the max capacity) and if so spawn the enemy somewhere along a circle surrounding the player (you choose a reasonable radius for the circle)

Levelling System :

- Each time the player picks up an XP item, the player will gain one XP point
- When the player accumulates enough XP points to level up, they should be presented with a 'level up' user interface with two random options of stats to increase
- You choose which stats to make upgradeable, but you must support at least 4 unique stats, which will affect the corresponding gameplay mechanics (e.g., speed upgrade should make player move faster)
- The required number of points to level up should increase following an exponential function (there are many online articles and tutorials discussing this topic. Ask TA or instructor if you are unsure where to begin).

User Interfaces

In addition to the required user interfaces in the general requirements, there are a few additional user interfaces requires:

Level Up : A screen that appears when the player levels up, giving the player an option for a permanent stat boost

Stats Display : There should be a UI element somewhere on the screen to display the current value of upgradeable stats

Game Over : When the player runs out of health, a game over screen should appear, with the option to either start a new game or return to main menu.

HUD : A headsup display with bars in the top-right corner of the screen showing player's current health, and the player's progress to levelling up

Harvest 306

Unlike the 306 survivor, this game is more focused on data structures, architecture, and design patterns. There is less going on in terms of player interaction and front-end, but there is a decent amount of data structures and modelling that need to go into the project.

Art

Tiles : There should be three tiles: grass, dry dirt, wet dirt. Sprites are supplied for these
Crops : A spritesheet is supplied with many different choices for crops and multiple growth stages
User Interface : A spritesheet is supplied containing an assortment of ui elements that can be used for menus and hud

Audio

The following are requirements for Music/SFX. A link to a collection of game assets is included in the starter assets, but you are free to find your own audio, if you prefer (as long as it is legally required with a license).

- Clicking on a UI button (i.e., Main Menu, Pause Screen, Toolbag)
- Opening/closing a menu (i.e., Options menu, Pause menu)
- Background music should be playing on loop in the Main menu and in the game
- A sound should play when the player interacts with a grid space (i.e., waters a tile, hoes a tile, plants a crop, harvests a crop)
- A sound should play when the player switches tools
- A sound should play when a crop grows a stage

Mechanics/Rules

Tool Menu :

- The game should support three tools: Shovel, Watering Can, Hoe
- One tool can be selected at a time (the Shovel is selected by default)
- The selected tool should be highlighted in some way
- The mouse scroll wheel should be used to select the next/previous tool
- Additionally, the tools should be mapped to the hotkeys:

  Shovel -> `1`
  Watering Can -> `2`
  Hoe -> `3`

- The Cursor should change to reflect the currently selected tool
- When the player clicks on a tile, the currently selected tool will be 'used' on the tile (effects are described below)

Tiles :

- Three tiles are supported by the game: GRASS, DRY DIRT, WET DIRT
- Tiles should have a saturation level (i.e., how much moisture they currntly have)
- When a tile is watered, it should increase its saturation level by a specified amount
- GRASS tiles should always have a saturation level of 0 (i.e, watering it shouldn't do anything)
- Each tick, a tile should lose a point of saturation (cannot go below 0)
- A DRY DIRT tile has a saturation level of 0

- When a DRY DIRT tile becomes saturated, it should change into a WET DIRT tile
- When a WET DIRT tile desaturates down to 0, it should change into a DRY DIRT tile
- A WET DIRT tile should have a maximum saturation level that cannot be exceeded

Hoe :

- When selected, the Hoe will turn a `GRASS` tile into a `DRY DIRT` tile when used
- When selected, the Hoe will harvest a crop, if it is ripe

Watering Can :

- When selected, the watering can will apply an amount of saturation to a DRY DIRT or WET DIRT tile when used

Shovel :

- When selected, the `Seed Bag` menu should appear, with the option to select which seed to plant
- With a seed selected, the Shovel can be used to plant a crop in either a DRY DIRT or WET DIRT tile
- For simplicity, we will assume the player can plant a seed with no cost
- When a different tool is selected, the Seed Bag menu should disappear

Crops :

- Each crop has a specific number of growth stages (e.g., 3)
- Each growth stage has a specific number of ticks before it advances to the next growth stage
- A crop gains a tick if its tile is Saturated (i.e., WET DIRT). If the tile is DRY DIRT, the crop will pause its growth until the tile becomes saturated
- When the crop reaches its final growth stage, the crop can be harvested with the Shovel tool
- A harvested crop will leave the tile empty, ready for another seed to be planted with the Hoe
- When a crop is harvested, the player will receive a certain amount of money (different for each crop type)

User Interfaces

Seed Bag :

- The seed bag menu should contain a grid of 9 buttons (3x3)
- For this assignment, the top three buttons should be used to select a crop (3 unique crops supported)
- The remaining 6 buttons should be inactive with no interaction
- When one of the crops is selected in the seed bag, that should determine which crop is planted (see above for planting mechanics)
- The seed bag should 'remember' the currently selected crop (i.e., when the seed bag is closed and reopened the same crop should be selected)

HUD :

- There should be a label somewhere on the screen (probably top-right corner) displaying the amount of money the player has currently earned from harvesting crops
- The list of tool icons should be in the bottom center of the screen (see reference video on Canvas)
- The currently selected tool should be highlighted in some way

Evaluation

The evaluation for each project will be slightly different. The markers will be given a checklist with all of the features listed above, and the mark will be the number of features that are implemented correctly. Markers reserve the right to inspect code and request access to git repositories.

While there are marks for the presence of art and audio, there are no marks in this milestone for the *quality* of these assets. As such, it is advised to not spend much time on asset creation for this phase and to instead focus on architecture, data structures, and prototyping mechanics.

Provided Files

A collection of assets will be provided on Canvas for groups to use as they see fit to complete the project.

What to Hand In

One member of each group should hand in a single .zip file containing the following:

- The entire godot source code directory
- A .txt file with a link to the itch.io page containing a web build of the game (this is what the markers will be playing to mark the game, make sure everything works on the web platform).
- A gitlog containing all of the commits for the entire milestone.

In the (likely) event that the submission is too large to submit to canvas, a single .pdf or .txt file should be submitted with a link to a cloud platform (i.e., OneDrive, Google Docs, Dropbox, Github, etc). Make sure it is configured so anyone with the link can download the materials, marks may be deducted if markers have to contact you to gain access.