Golem Grudge FGP22 - README

The game was created as a part of the Game Project 1 course at Future Games. My ambition with this project is to pass with distinction (VG).

These are the criteria that I have aimed for in particular:

- General
    - (G) Only play scene is required

        The game has two scenes - a main menu scene and a play scene.

    - (VG, small) Add main menu (start) scene and game over scene

        The navigation through the scenes is done with the help of the UnityEngine.SceneManagement namespace. The default methods like LoadScene() and ExitApplication() are called from the User Interface. I decided to put the "Game Over" text in the play scene so that the player is not kicked out of their current game session, and instead can choose whether they want to continue playing or exit the game.

        Pausing the game is also possible, and uses global variables like AudioListener.volume = 0 and Time.timeScale = 0 which can be accessed through MonoBehavior.

- Turn based game
    - (G) You can have two players using the same input device taking turns.

        The player uses the mouse and keyboard to control their Golems. The turns are swapped with the help of a Game Manager that updates after an action is performed or the timer runs out.

    - (VG, large) Support up to 4 players (using the same input device taking turns)

        The player can decide in the Main Menu how many players should be present. The valid intervals are between 2-4 players, which spawns the appropriate teams based on the players request. The same game manager handles the player's turns through an integer, and will adapt itself to the amount of players left.

        For example, if Green's Golems have been eliminated, they will no longer be given a turn, and it will jump to the next player in the index.

All golems (4 teams of 4 golems each) are active and sorted into lists by default through the Initialize, and the appropriate teams will then be disabled based on the player’s wishes.

- Terrain
    - (G) Basic Unity terrain or primitives will suffice for a level

    The map is constructed in ProBuilder with the help of some free material assets from the Asset Store.

- Player
    - (G) A player only controls one worm
    - (G) Use the built in Character Controller. Add jumping.
    - (G) Has hit points
    - (VG, small) Implement a custom character controller to control the movement of the worm.
    - (VG, small) A worm can only move a certain range
    - (VG, medium) A player controls a team of (multiple worms)

    Each active player controls a team of 4 Golems that the player can switch between. This is handled by the Game Manager which updates the currently selected golem in each player’s list. By default, the first Golem in the list will be selected once that player’s turn starts, but the player can navigate freely by pressing “TAB” during the game.

    Each Golem has properties that are unique to them, such as their Health and name tag. The golems do not use the character controller component present in unity, and instead uses the Rigidbody component to handle movement. Through input by the player, forces are added upon the golems, whether it is walking, jumping or explosions affecting them.

    Golems can also Hover each turn, giving them a Jetpack-esque movement boost wherever they might need it. Due to the map’s layout, it felt like a great tool to give some extra control to the player.

    When the player is moving (the movement vector != 0) a variable called distanceMoved increases. When that variable reaches the desired limit, that golem in particular cannot move any further during that turn. The player can, however, still perform actions like shooting or punching, or switching to another golem should they want to move any other character during their turn. When the turn is over, these temporary booleans and conditions are reset so that the player can hover and move again the next turn.

- Camera
    - (G) Focus camera on active player
    - (VG, small) Camera movement

The camera is controlled with the help of the Cinemachine package in Unity. Two separate cameras are present inside the scene, and take in the current player's transform.position and transform.rotation.

This creates a cinematic feel when the players are switched around. The second camera in particular is only active when the player zooms in during a Rock Throw, and helps the player make fine adjustments to their shot.

The scripts that use the camera, such as the Rock and the Game Manager simply give the transforms to the camera directly when it is needed.

- Weapon system
  - (G) Minimum of two different weapons/attacks, can be of similar functionality, can be bound to an individual button, like weapon 1 is left mouse button and weapon 2 is right mouse button

    Each Golem has 2 different attacks, bound to different buttons. The primary weapon in the game is the Rock Throw, which instantiates a projectile and adds a force to it. This projectile responds to collision with the environment or a player, and detonates on impact, dealing damage to golems based on their proximity to the explosion. This weapon can also deal damage to golem firing it.

    The second ability is a Smash attack which requires Golems to be adjacent to its target. The Golem will then punch all Golems in front of the player, dealing damage and sending them flying in whatever direction the player was aiming at.

Created by William Michelson FGP22