CS 3160 – Game Programming

Project 4 – AI and UI

Learning Objectives

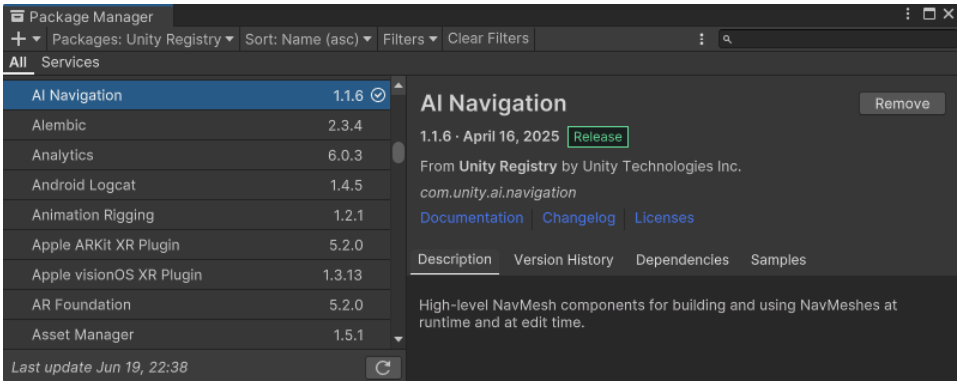
- Learn how to implement a finite state machine, main menu, and health bar.

Overview

In this project you will become familiar with creating an AI zombie NPC logic and UI elements.

Sample Code

To complete this project some sample code is required. This is provided in the form of a Unity Package. Download the "SampleCode-Project4" Unity Package from Pilot. First, create an empty Unity project using the "3D (Built-In Render Pipeline)" template. Then, go to "Assets" > "Import Package" > "Custom Package…" then locate "SampleCode-Project4.unitypackage" downloaded from Pilot and import all files. After this, you must install the AI Navigation package provided by Unity here:

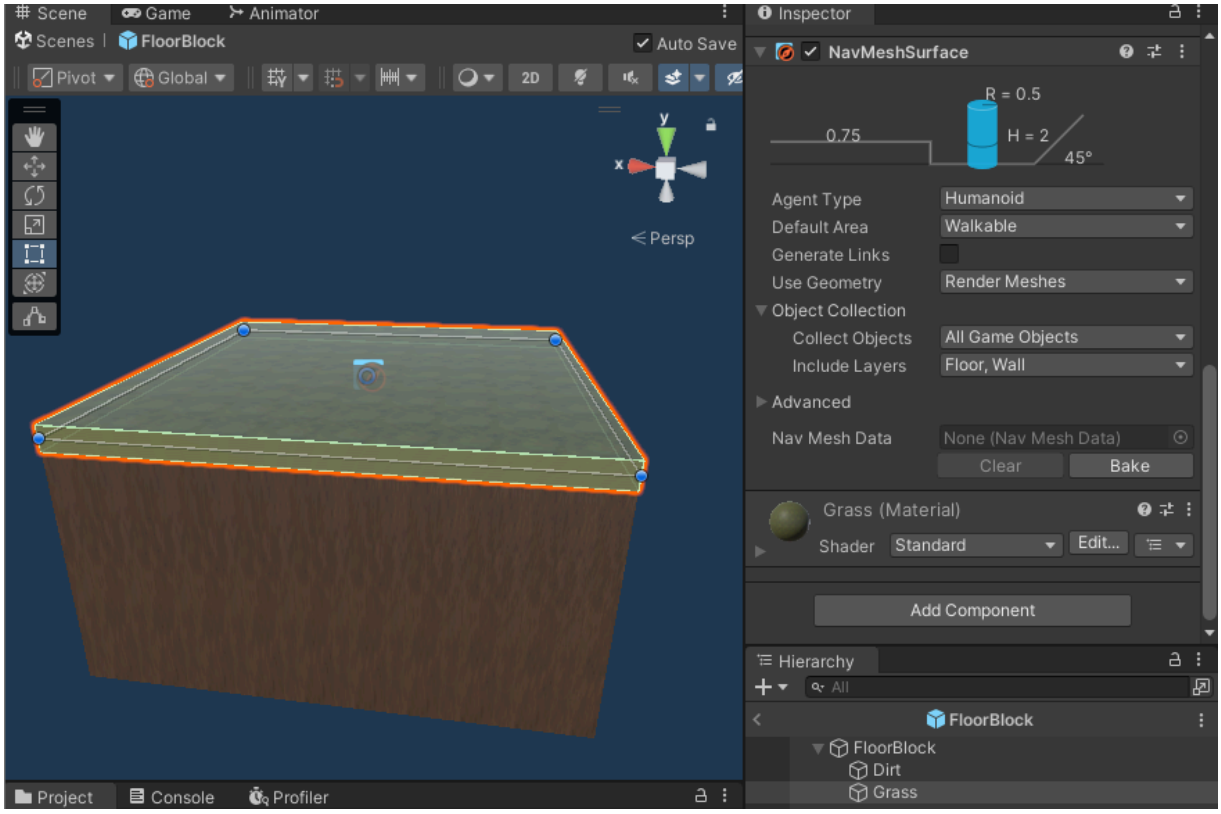


If importing was successful you should be able to load the "Project 4 - Level Easy" Scene in the "Scenes" folder and you can test the game.

NOTE: Importing a Unity Package does not import Layers. The layers used are listed above.

Modify Floor Block

Modify the floor block prefab to include a “NavMeshSurface” on the grass surface. Also, make the NavMeshSurface include ONLY the layers “Floor” and “Walls”. Do not bake the Nav Mesh Data in the prefab view, only bake the Nav Mesh Data in the Scene View.



Zombie Finite State Machine

Some starter code is provided to implement finite state machines for characters in this game. The ZombieStateMachine class inherits from StateMachine and can transition between States. Some states are already partially implemented, you will have to implement the following:

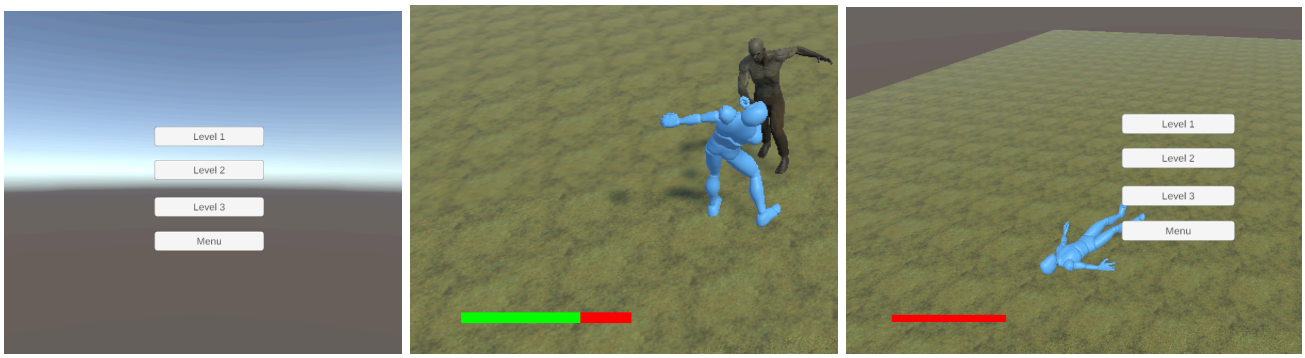
- Wander State
    - Entering state: set the speed of the NavMeshAgent to “wanderSpeed” provided in ZombieStateMachine
    - Executing state: periodically set a random new position for the NavMeshAgent and use the magnitude of the NavMeshAgent’s desired velocity to set the animator “isMoving” bool
        - Hint: to periodically check for something use:
          if (Time.time - lastPositionSelectedTime > zombieStateMachine.wanderDuration)
        - Hint: to get a random position use:
          navMeshAgent.transform.position + new Vector3(Random.Range(-wanderDistance, wanderDistance), 0f, Random.Range(-wanderDistance, wanderDistance));
    - Exiting state: none
    - Transitions to:
        - Death State if is not alive aka dead
        - Chase State if the player is within the chase distance and player is alive
            - Hint: you can check distance using:
              Vector3.Distance(transform.position, player.transform.position)
        - Attack State if the player is within the attack distance and player is alive
- Chase State
    - Entering state: set the speed of the NavMeshAgent to “chaseSpeed” provided in ZombieStateMachine and animator bool “isChasing” to true
    - Executing state: set destination of NavMeshAgent to player’s position and use the magnitude of the NavMeshAgent’s desired velocity to set the animator “isMoving” bool
    - Exiting state: set the animator bool “isChasing” to false
    - Transitions to:
        - Death State if is not alive aka dead
        - Wander State if the player is outside the chase distance or player is dead
        - Attack State if the player is within the attack distance and player is alive
- Attack State
    - Entering state: set the animator trigger “Attack”
    - Executing state: none
    - Exiting state: if the player is within our attack distance when we exit the attack state, do damage to the player
    - Transitions to:
        - Death State if is not alive aka dead
        - Attack State if the the attack duration time has passed and the player is within the attack distance and player is alive
        - Wander State if the “attackDuration” has passed since entering the Attack State
- Death State
    - Entering state: set the animator trigger “Dead” and the NavMeshAgent speed to 0
    - Executing state: none
    - Exiting state: none
    - Transitions to:
        - none

Menu and Health UI
A MenuController script is provided but you must implement 4 methods:

- public void LoadLevel1()
    - Loads Scene “Project 4 - Level Easy”
- public void LoadLevel2()
    - Loads Scene “Project 4 - Level Medium”
- public void LoadLevel3()
    - Loads Scene “Project 4 - Level Hard”
- public void LoadTitleScreen();
    - Loads Scene “Project 4 - Title Screen”

Then, using a prefab, call these methods from a menu UI on the Scene “Project 4 - Title Screen”. Also, in Scenes other than the title screen, use the Escape key to open up a menu that allows you to return to the title screen. If this menu is open unlock the cursor, if this menu is closed, lock the cursor.

In Scenes other than the title screen, add a health bar UI element that displays the players health. The health bar must show the changes to the player’s health if they take damage from the zombies. To allow for tweaking the psychological feel of our game, do not display the health as just (currentHealth / maxHealth), instead manipulate the value to increase the player’s tension in some way. For example, displaying $(currentHealth / maxHealth)^3$ will make the health bar display 0.1% health remaining even though the player has 10% health. An example of the UI elements are shown below:

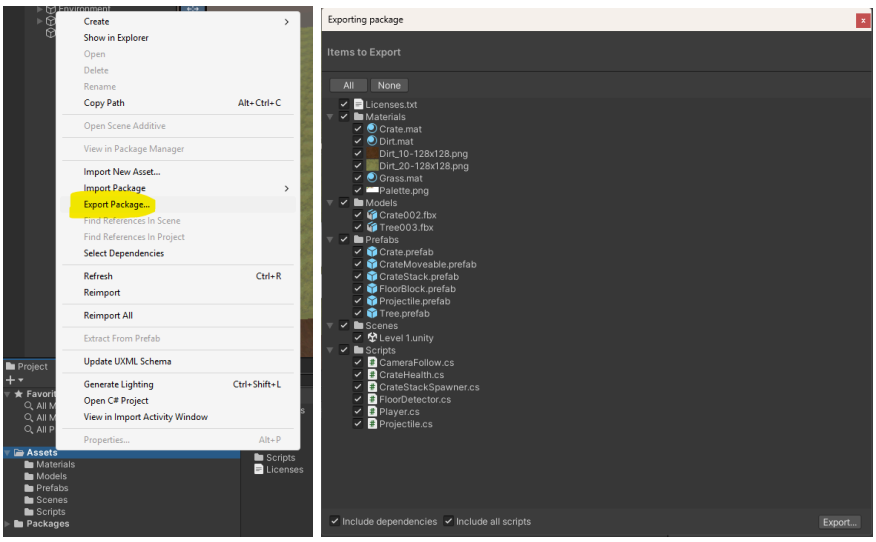


Example Solution
If you want to see what your final project should implement, please try the following demo:

https://maxgilson.itch.io/zombie-ai-demo

How to Submit

To submit your code, you must pack your project into a Unity Package. To create a Unity Package first right click on the “Assets” folder inside of the Unity Editor, select all items and hit Export. Lastly, name your file “YourLastName-Project4” where “YourLastName” is your actual last name, and submit this file to the Pilot dropbox:



Grading

This lab is worth 5.00 points, distributed as follows:

Task	Points
There are no build or runtime errors	1.00
Zombie FSM transitions implemented	1.00
Zombie FSM states	1.00
Menu UI implemented	1.00
Health Bar implemented	1.00
Total	5.00