Assignment 6

CS 381: Game Engine Architecture
Spring 2022
Max Score: 100

Objectives

- Demonstrate (again) that you can identify, formulate, analyze,and solve complex game engine architectural problems by applying principles of computing, engineering, science, and mathematics.
- Demonstrate (again) that you can apply computer science theory and software development fundamentals to design and integrate the components of a game engine and build a scene with interacting movable entities.

Assignment objectives

- Demonstrate that you can learn Unity quickly having studied the architecture common to all game engines and having implemented significant portions of a game engine using this architecture.
- Demonstrate that you know the mathematics behind simple game artificial intelligence techniques and can implement them in a game engine.

Assignment

Design and implement mouse selection and simple AI commands for entities in your Unity game engine. You will design and implement three commands: move, follow, and predictive-intercept. To do this you need to design and implement

- a command class and subclasses
- a unit AI aspect that manages sequences of commands, and
- subclasses for move, follow, and predictive-intercept commands.

Use my solution to assignment five, on Canvas, as your starting point.

Note that in this assignment, the left mouse button selects the entity under the mouse. The right mouse button commands selected entities. Pressing the shift key while right mouse clicking *adds* a command to the list of commands for the selected entity.

AI Part 1

Simple AI (70 points)

You will design and implement an AI aspect (script attached to entities in Unity) named UnitAI which manages a sequence of commands that are executed in order. Commands are of three types: **Move**, **Follow** and **Intercept**.

Essentially, if you right click your mouse button over an entity, you will choose Follow or Intercept. If there is no entity under your mouse when you right click the button, you will choose Move. You distinguish between Follow and Intercept by checking whether the left side ctrl key on your keyboard was also being pressed when right-clicking over an entity.

These three behaviors are the basis for much more intelligent entity behavior. For example, executing a sequence of Moves enables an entity to follow a path, patrol an area, or attack another entity from a different angle.

Move (15 points)

Move moves the currently selected entity to the location (move location) right-button-clicked by your mouse. **This location must be over open water.**

Follow (15 points)

When you right-button-click over another (target) entity, the currently selected entity **Follows** this target entity at a **relative** fixed offset of $(100, 0, 0)$ with respect to the target entity. In Unity's coordinate system this means your follow position is 100 units to the right (starboard) of the target.

Intercept (20 points)

When you press the left ctrl key while right-button-clicking over another entity (the target), the currently selected entity predictively intercepts this target entity by computing and aiming for where the target will be.

UnitAI: Command sequencing (20 points)

Finally, when you press the the Left-Shift key while pressing the right mouse button, the command (one of the three above) is added to the list of commands maintained by UnitAI. When a command is done, the UnitAI aspect executes the next command in the list or does nothing if there are no commands to execute.

UnitAI inherits from Unity's MonoBehavior and maintains a list of `commands` of type `Commands`. You will implement the following additional methods in UnitAI

1. `void SetCommand(Command c)` to clear all current commands and set the first command in the list to `c`.

2. `void AddCommand(Command c)` to add `c` to the end of the current list of commands. The player presses the LEFT SHIFT key and the right mouse button to add a command to the list.

Note again that the UnitAI class keeps a `List<Command> commands` of commands. AddCommand adds to the end of this list (first in first out) and SetCommand clears the list and places the new command as the only member of this list.

A command inherits from the Command class and can be one of Move, Follow, or Intercept. The command class hierarchy **does not** inherit from MonoBehavior. Commands implement a public constructor, and the following methods

1. `public void Init()`. Initializes the command if needed.

2. `public void Tick()`. Called from UnityAI to perform per frame processing.

3. `public bool isDone()`. Called from UnitAI to check if the command has finished. Move finishes when the selected entity gets within a small distance of the commanded move location. Follow never finishes. Intercept is done when the selected entity gets within a small distance of the target. UnitAI removes the current command from the list and starts executing the next command in the list when the current command is done (that is, when `isDone` returns `true`).

4. `public void Stop()`. Stops the command and must be called before UnitAI removes this command from the list of commands.

AI Part 2. AIMgr: UI for AI commands (5 points)

The AIMgr is a globally accessible manager class that inherits from MonoBehavior. It handles right mouse clicks and the two associated keyboard modifiers: Left Shift and Left Ctrl. I provide my AIMgr code in AIMgr.cs.

Mouse Selection (20 points)

You will modify your SelectionMgr to handle left-click-mouse selection. Left clicking your mouse over an entity, selects the entity under your mouse and places a correctly scaled selection circle under/around this entity. You may download and customize a selection circle from Canvas under Files → UnityPackages. Play with the assignment webGL export for what this might look like in my solution. Note the command decorations are optional for this assignment.

UIMgr for entity information (5 points)

Design and implement a UIMgr that manages a grid of text elements that displays an entity's name, speed, desired speed, heading, and desired heading.

Camera control (0 points)

Same as assignment 5.

Quitting (0 points)

Hitting the escape key should shut down your running game engine.

Design Constraints

You must have at least five ships in your scene with each ship having different ship properties.[1] Please build on top of my solution to assignment 5 although this is not required. The TAs and I are happy to explain every aspect of my solution to assignment 5.

Extra Credit

- Add flying entities (easy)
- Add group mouse selection and commands apply to all selected entities (Medium)
- Add a LineMgr that helps create and parent Unity LineRenderers to visualize an entity's command sequence (Medium).
- Implement scenic islands through which we can cruise (medium)
- Add background music (medium)
- Add wakes to all entities (Hard to do well)

Turning in your assignment

Assume that this format will be used for all your laboratory assignments throughout the semester unless otherwise specified.

1. **Make a movie of your working demo and save this movie file in** `as6Movie.mp4`. The movie should demonstrate each of the features required by the assignment.

[1] At least different acceleration and turn rate's

2. Before lab, upload your code and movie file to Canvas. You will zip or, tar and gzip, the folder with all your code and movie into one file. Upload this one file on Canvas.

3. Demonstrate your working program in the lab on the due date. Demonstrate each of the above features (and any extra credit). You will be graded on your in-class demonstration.

4. Your FULL name and email address, should be on all submitted files.

Ask on Piazza if you have any questions.