Wiz II

Due: October 15, 2019

Introduction

This week, we’re finishing Wiz with AI and pathfinding.

Design Check

- Which AI framework will you use? What engine-side classes do you need to implement this? What behaviors will you need for your game objects to interface with those classes? When will the AI code actually be called?
- How will pathfinding be integrated into your engine? How will your AI make use of pathfinding?

Playtesting Requirements

- Your handin must meet all global requirement.
- Your handin only crashes under exceptional circumstances (edge cases)
- You game should have a map containing passable and impassable tiles.
- Your game must have a unity that can be controlled by the player.
- Your game must have at least one enemy unit that moves around deterministically (i.e., the same actions by the player result in the same enemy behavior). There must be a visible reaction when the player and item collide.
- The player-controlled unit must not be able to leave the map.

Primary Requirements

Primary Engine Requirements

- Your engine must implement an AI library using either behavior trees or goal-oriented action planning.
- If using behavior trees, this means that you must include a behavior tree, selector, and sequence class. You must also include a node and action/condition interface.
- If using GOAP, this means that you must include a game state class, an action class/interface, and a condition class/interface. You must be able to search over and action set using a predefined start state and goal condition.
- In both cases, your engine must have an A* implementation.

Primary Game Requirements
• Your handin must meet all playtesting requirements.
• The enemy unit should use your engine’s AI framework. All of the AI tools included in the engine requirements should be used when constructing your AI. All of the behaviors defined for your enemy should be visible at some point when playing the game.
• It must be possible to start a new game without restarting the program.
• Your game never crashes.

Secondary Requirements

Secondary Engine Requirements

• Your engine must meet all primary engine requirements.

Secondary Game Requirements

• Your game must meet all primary game requirements.
• The enemy unit should move according to a path generated using A*.
• Your game must meet at least two of the extra game requirements (that weren’t used for last week’s game requirements).

Handing In

Hand in the entire directory tree for your project, including both your engine and game code. You must also include a README file that describes how to verify each requirement and an INSTRUCTIONS file that describes how to play your game, as specified in the Global Requirements. If you’re using Eclipse execute cs1971_handin wiz2 from the root directory of your Eclipse project (something like /path/to/a/workspace/tic). If you’re not using Eclipse, the directory you hand in must include an ant build script that compiles your code into an executable jar.