PROJECT 4: GAME ENGINE JAM

Due: In-class demonstration 11:30 am - 2 pm, December 16, 2022; code delivered to handin by end of day, December 16, 2022

DESCRIPTION:

This is the big cheese, the big kahuna, the big show, the big <insert hyperbolic metaphor here>. You have 2.5 weeks to put this assignment together as a team. The teams were previously established and may not be changed unless approved by the instructor.

This project is worth **35 points**, divided up broadly into three categories:

1. Does it work and are the specific requirements of the assignment implemented? (15 points)
2. Was the demonstration informative about the strengths and weaknesses of your project? (10 points)
3. Is the code clean, organized, and demonstrative of pride of craftsmanship, and easy to execute? (10 points)

On December 16, you and your team will submit this project for grading in two ways:

1. In-class demonstration: You will have 10 minutes in front of the entire class to demonstrate your game playing and show how you implemented the required engine updates. This is your chance for your team to show off your game and game engine in its best light.
2. By the end of the day, December 16, you must submit the code for your project to the SoC handin system. The specifics for doing this are posted on the website for the class. Pay careful attention to this task: many students who were very competent have messed up this task and suffered point loss. We will expect to be able to run the game on our own machines, as well as inspect the code.

GAME ENGINE

As we have discussed in class, our python-based game engine must have the following organization of directories (gamey is the name of my game engine code - feel free to give yours any name that you like, while retaining this directory organization inside):

```plaintext
gamey
├── assets
│   └── sounds
│       └── textures
├── engine
│   ├── actor
│   │   └── action
│   │       └── entity
│   ├── asset
│   │   └── action
│   │       └── entity
│   ├── crowd
│   │   └── action
│   │       └── entity
│   ├── enviro
│   │   └── action
│   │       └── entity
│   ├── fx
│   │   └── action
│   │       └── entity
│   └── physics
│       └── action
│           └── entity
│   └── play
│       └── action
│           └── entity
│   └── render
│       └── action
│           └── entity
│   └── sound
│       └── action
│           └── entity
│   └── story
│       └── action
│           └── entity
│   └── ui
│       └── action
│           └── entity
│   └── utility
│       └── action
│           └── entity
└── utility
```
Do not forget that inside each directory, there needs to be a valid python file called "__init__.py". An empty file is valid, but you can also use it to create utility or factor functions, for example.

Final word of advice, paraphrasing Gandalf: “All we can do with our time in this world is build the best possible game engine”.

GAME JAM THEME

In addition to the graded work on the game engine, we are also conducting an informal game jam competition. This will be judged by an independent panel of experts, based solely on the game play demonstrated by each team during the final exam period on December 16, 2022. The judges will decide the criteria and the award process. This game jam is for fun. It has no bearing on the grading. The theme for the game jam is:

CONTEMPLATION

GAME ENGINE ASSIGNMENT

In this assignment, you must develop new tools/features/capabilities within your game engine. This new game engine development will allow games to have multiple levels of play. A level is a completely playable set of entities and actions that run through the game loop and engage in game play. Only one level is present in memory at a time. When a user advances to a new level, the game must remove the existing level (e.g. via python garbage collection), load a new level, and commence play with the new level. The game engine must have entities and actions that combine to accomplish this level description and loading. This is the new feature(s) required in your engine.

The game must be able to load a minimum of 23 levels. Each level must be more complicated and/or difficult to play than the previous level.

At any time, pressing the escape key or the quit button on the window should end the game and python should terminate.

The game code itself should be contained in the “python” directory, called “goat.py”, as a python script that imports and uses the various elements of the game engine to initialize the game, entities, actions, and then runs the game loop.

The game must also have a credit screen, listing the members of the team.

STUDENTS IN 6160:

If you are in 6160, there are additional implementation elements. If you are in 4160, you may implement these also because you may find them useful. But 4160 grades will not be impacted by this additional detail.

With each level, you must include a timer that tracks how long the level is in play for moving on to the next level. When play at a level is complete, the timer prints to stdout the amount of time, in seconds, that the level was in play.