Updates

Space Adventure

Team: Cindy (Xinzhu) Fang, Jenny Kim, Justin Warring
May 2, 2018

Our very first idea for this project was a spiking neural network (SNN) which had certain challenges that made it difficult for us to implement within the amount of time we had and with the level of sophistication we actually wanted. We thus decided to work on a different project that involved the mish-mash of two old school Atari games, snake and asteroids.

Our design initially started as a various mix of classes and subclasses, most of which we ended up keeping such as asteroid, debris and ship. Starting this project with set classes in mind helped with the design process, allowing us to draw relationships between objects (such as subclassing) so that we didn't have to rewrite code or redefine variables in similar classes. However, it got to a point where we had a great number of separate classes, each with its own relationship to another, which started to get difficult to keep track of.

We ended up keeping much of the classes that we had before but made certain changes that helped reduce the number of classes and parent classes we had to generate. For instance, instead of having two separate parent classes of ship and sphere, we ended up removing the ship parent class altogether and instead extended Sphere for ship. This helped reduce the various types of relationships between classes that we had to keep track of, which helped in organizing and assigning tasks to group members.

We similarly restructured the methods inside the world class, where we organized the draw methods into one larger method. This allowed for, again, an easier splitting of tasks between group members and a clearer reading of code.

We also kept in mind the various functionalities that we felt a player would want within a game, thus leading to the creation of a life counter in the top left corner that actively kept track of the player's lives and also an exit, a pause, and an unpause button at the top center of the screen. This would allow for players to pause a game midway for perhaps a break and then restart at their leisure.

The opening window also gives the player the full instructions necessary upon running the program but also checks with the player to see if they want to truly play the game or cancel out.

Our keyboard inputs were also specified to allow for full range of movement, allowing the player to not only shoot while moving but also move diagonally (and also shoot) and rotate.

We also originally had set up difficulty to be set at "levels" that the player could input. However, we felt that this was not as challenging for the player and instead designed the difficulty so that it would exponentially increase as the game went on but at a random pace. This

would keep the player more mentally stimulated, which we feel is part of what makes a game fun.