**STATUS REPORT #5**

**Scott & Nicole**

List of scratches

AH Animation Class: Has an AniSprite class that animates and a Hamster class that grabs the texture from the screen based on the choice of the user and uses the direction instructions to move the mouse; the hit detection of the hamsters are also fixed using a function in the hamster class, called getThisRect

AH Level 4: Created a new level to add something new and interesting with the system. In this scratch after creating the level I went and added a timer system for all the levels and also added a Sprite in the bottom corner of the screen to show which mouse a certain player looks like.

AH Main Menu Fun: Made to add more of an arcady feel while in the main menu. The hamster just goes around the board. I plan to do something more with this but for now this is fine to show the purpose

AH Obstacles: The main feature of Level 3 is obstacles randomly placed throughout the screen, inside the walls. Two obstacles are created as an array, using the Wall class. They are then placed randomly at the start of the level and the player then has to deal with them being there.

Release Version 5.0 integrates everything from these scratches, as well as some more cleaning up and fixing of bugs.

Major challenges/setbacks

The animation class took a long time to create and was difficult. We created two classes. AniSprite creates a sprite and animates it. Hamster grabs the texture from the screen from the AnimalChoice screens, based on the choice of which colour hamster the player wants; it also grabs direction instructions from the game screens and updates the hamster’s position and coordinates accordingly. In the game screens, a new AniSprite is created, as well as a Hamster. Then, the aniSprite calls animate(). Then, the hamster calls move(), which passes the direction instructions, as well as size and speed coordinates based on if the hamster has moved over a pellet or poison. Then the hamster calls animation(), using nFrame. Then the wall is checked for hit detection, calling an isHitS() function in the hamster class. If the wall is hit, the hamster calls outOfBounds() to set it back inside the wall. Then spTemp from the hamster class is drawn. If the game is over, reset() is called to reset all of the coordinates, speed, and size variables. This was quite a challenge and took up so much time, but it was a very interesting challenge to overcome and really helped me with the understanding of classes, functions, and passing information and variables.

A more minor setback was dealing with the more finicky parts of the code and trying to make sure that everything placed correctly and won’t interfere.

Source any web site/book that helped you with that concept

Thanks to the Radioactive Hamsters team for letting us take a look at their code for the animation class. It helped a lot with understanding the general structure of what is would look like.

General information:

ICS3UI website

Lessons learned from the last two weeks:

1. GitKraken helps so much! It resolves the problems that always came up with binary files and lets you know if you need to pull, before it allows you to push and create merge problems.
2. Functions and classes help so much with cleaning up code and allows you to isolate certain aspects and features of something. It was very interesting to learn how to pass information and call functions.
3. Making sure to keep all unneeded code out of the program is very important to make sure it doesn’t mess with what you are trying to put in, but also make it easier to find everything.