Franton Lin & Kai Levy
Software Design, Spring 2015
3/12/15

Mini Project 4 Writeup

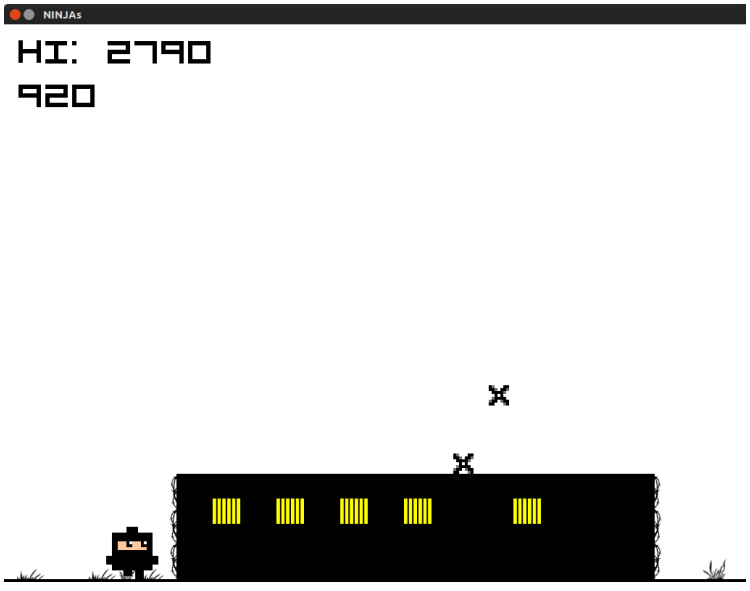
Project Overview [Maximum 100 words]
Write a short abstract describing your project.

With this project, we created an infinite runner platforming game that features an 8-bit Ninja character. The game takes inputs in the form of key presses from the player, and controls the Ninja. The goal of the game is to score as high as possible, running far and picking up thrown shurikens while avoiding death.

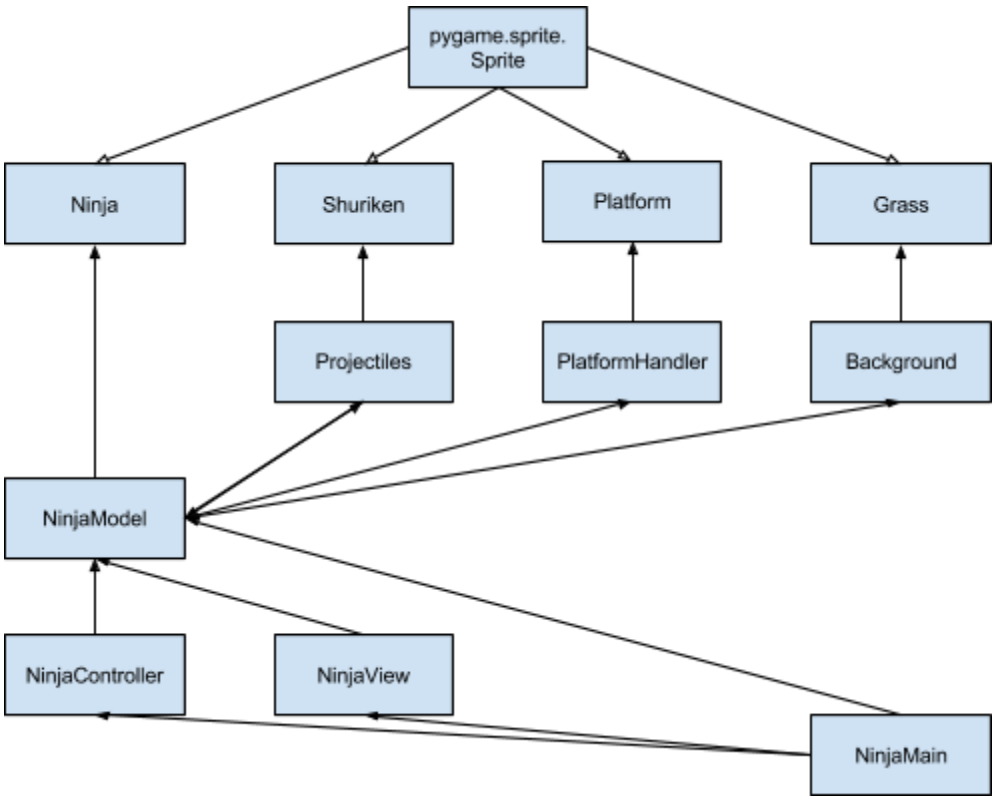
Results [~2-3 paragraphs + figures/examples]
Present what you accomplished. This will be different for each project, but screenshots are likely to be helpful.

With our project, we managed to make what we hope is an entertaining and visually appealing game. The sprite for the Ninja is constantly updating to create the sense of running. Similarly, the objects and background constantly flow to the left to enhance that effect. The player controls the Ninja character to avoid obstacles while running, and the score reflects the Ninja's progress.

The obstacles and other decorations are generated randomly, to make gameplay interesting and potentially infinite. Each obstacle calculates collisions with the character and every other obstacle. The character can run on top of every platform. If a shuriken hits the character or the character runs into a wall, the game is over. If a shuriken hits a platform, it loses its power, and the character can collect it for additional points. Additionally, the game gets harder as the score increases.



Implementation [~2-3 paragraphs + UML diagram]
Describe your implementation at a system architecture level. Include a UML class diagram, and talk about the major components, algorithms, data structures and how they fit together. You should also discuss at least one design decision where you had to choose between multiple alternatives, and explain why you made the choice you did.



In our implementation, we had all of the game objects inherit from pygame.sprite.Sprite. This makes it extremely easy to put the game objects into groups, allowing us to update and draw them in groups. The groups are handled by the handling classes (Projectiles, PlatformHandler, and Background). We used the model-view-controller paradigm to isolate different tasks of the program to NinjaModel, NinjaController, and NinjaView. The NinjaMain class initializes, connects, and loops the model view controller.

One design decision we made was to add handler classes to keep track of groups of one type of object. We could have just kept lists of the groups in the NinjaModel class, but creating the handling classes made the code more readable and organized. It also made debugging slightly easier.

Reflection [~2 paragraphs]
From a process point of view, what went well? What could you improve? Other possible reflection topics: Was your project appropriately scoped? Did you have a good plan for unit

testing? How will you use what you learned going forward? What do you wish you knew before you started that would have helped you succeed?

Our project was overall successful because we were able to make a simple and entertaining game in the time frame. It looks very appealing, with clean art in an 8-bit style, which was one of our goals. There are still a few bugs that don't adversely affect the gameplay experience, that we would fix if we had more time. Additionally, since this is our first foray into pygame, the code is somewhat messy and much less efficient than it could be. The collisions were a huge pain to deal with and could still be improved.

For unit testing, we developed classes individually and ensured that they worked in a containerized version before developing them further. However, this method got a little bit messy when the different sprite objects had to interact heavily. While the interactions function, the structure of them could definitely be cleaner. Moving forward, we will make sure to plan our collision methods further in advance, and have appropriate methods for every interaction. We with that we had had a better understanding of the difficulty of collisions, as well as made a better overall game architecture before we had started.