Intro to Game Programming

Artemis' Diamond

Albert Chung & Yan Manelis

POLYTECHNIC INSTITUTE OF NYU

Artemis' Diamond is a single player RPG game. The player plays as a character named Artemis. Artemis is on a quest to find the diamond that has been stolen from him. The diamond is under the possession of Bob the Blob, ruler of the "OtherWorld." In the inverted "OtherWorld," the enemies are evil Artemis clones. They are the ones that have stolen the diamond and passed it along to their king. The goal of this game is to fight and eliminate all the evil Artemis'; afterwards you are finally able to encounter and destroy Bob the Blob to retrieve the diamond that is rightfully yours.

We kept the gameplay controls very simple. The directional arrow keys are used for character movement and action selection during battle. The enter button is intuitively used to continue and select actions. The escape key is used to exit battle scenes and at the completion of the game – to reset. As you start the game, the first two levels have simple on-screen tutorials to help you get acquainted with the gameplay and objective. You can move Artemis around and complete your objective. He is able to collect health regeneration potions around maps that include them and walking up to an enemy initiates a battle with them. To move on to the next level while playing the game, Artemis' goal is to eliminate every single enemy on the map. Each "OtherWorld" enemy is uniquely and familiarly named for comedic effect. If Artemis fails to defeat an enemy, the game will end and the player will have to start over from the beginning. When Artemis defeats all enemies as well as the final boss, Bob the Blob, the diamond is revealed and Artemis is able to collect it, thus winning the game. We included a little fun easter egg at the game-win screen where you are able to run around at super-speed.

The game code itself is interesting because it is an RPG style game utilizing a tile map system similar to a classic Nintendo game. Instead of perhaps using a .txt file like some of our classmates chose to do, we decided to hard code the maps, which can be seen in the tilemap.cs class. Each tile is defined to a specific tileID and rectangle on the tileset. The game has multiple scenes switching back and forth. There is a main menu, main game screen, battle screen, and a win/loss screen. We had an idea to use message boxes to further interact with the player, but discovered that the XNA API does not natively support windows message boxes. This was done to keep most things compatible with the XBOX console. After some research, we were able to tap into the windows API and implement the use of message boxes in our game for the Windows platform portion of XNA. The battle screen and combat system is activated when the player intersects an enemy. This is checked by rectangular collision detection. Upon loading the battle, the screen displays health bars, character names, health values, and relevant character sprites. The health bars are implemented using two rectangles, one for the background (gray) and one for the current health (red). Both are initialized to the same size, however, the red one is proportionate to your health over the amount of health you started the battle with. The use of health bars is nostalgic and similar to many classic games. The first strike is determined by a random generator that is initialized when the game is first started and is passed every time the battle screen starts with the combat system. It determines whether the enemy attacks first or the player attacks first. There is a battle delay implemented for the enemy attack so that the game does not have instantaneous decrease in player health when a player decides to attack and the enemy decides to attack right after. That was an issue we actually ran into early in development, where the enemy attack would happen instantaneously after the player attack.

Some difficult aspects were designing the levels with the tile system implemented. We would have to run every time we wanted to see changes in map. A level editor would be easier to see changes or by using an external level loading system using text files. There were a few bugs with the combat code with the enemy attacking twice and once again after the battle was finished but they were resolved.