Project Beta Report
Squirrel Quarrel: Protect Your Nuts Edition

Matt Coley
David Ickert

November 30, 2014

Contents

1 Summary of Milestones

1.1 Work Performed & Required Breakdown

Before beginning on the project, it became clear that many of our original labor divisions overlapped. For example, the original plan was for Matt to implement the map and for David to implement the movement and resctritions of that. However, this became a simple process for Matt to complete as part of the Map class development. We will attempt to maintain the division between tasks as originally planned, but as the project proceeds, it is easier to complete tasks on an "as needed and what is next" basis.

Below is a list of the original milestones. A ✓represent a completed tasks while an ✗is an incomplete task. New information is provided in bold.

1.1.1 First Milestone: Project Organization

Estimated Time: Half a day to one day.
Actual Time Used: About an hour.
Assigned To: David Ickert

- ✓ Create project. **(David Ickert)**
- ✓ Create empty files and folders based on the estimated UML diagrams. **(David Ickert)**
- ✓ Implement class inheritance. **(David Ickert)**
- ✓ Setup version control (such as GitHub). **(David Ickert)**

1.1.2 Second Milestone: Initial Implementation & Map

Estimated Time: 3 days to one week.
Actual Time Used: ???
Assigned To: Matt Coley

- ✓ Display the map. **(Matt Coley)**
- ✓ Pan the map. **(Matt Coley)**
- ✗ Place the home tree in the center of the map.
- ✗ Randomly place obstacles.
- ✗ Randomly place nuts (check for collision with obstacles).
- ✗ Randomly place enemies (check for collisions with obstacles and nuts).

1.1.3 Third Milestone: Sprites, Animations, and Basic Controls

Estimated Time: 3 days to one week.
Actual Time Used So Far: About 2-3 Hours.
Assigned To: David Ickert

- ✗ Replace the placeholders with sprites.
- ✓ Implement character animations. **(David Ickert)**
- ✓ Implement basic movement controls. **(Matt Coley as part of the map class)**
- ✓ Ensure restricted movement. **(Matt Coley as part of the map class)**

1.1.4 Fourth Milestone: Implement Menus & Refine Controls

Estimated Time: 3 to 5 days.
Assigned To: Matt Coley

- ✗ Create main menu & pause menu.
- ✗ Implement the functionality of the menus.
- ✗ Refine the controls.

1.1.5 Fifth Milestone: Player Logic: Actions & Combat

Estimated Time: One to two weeks.
Assigned To: David Ickert

- ✗ Implement the scoring system.
- ✗ Implement the powering up logic.
- ✗ Create collision detection for projectiles.
- ✗ Bust-a-nut special move.

1.1.6 Sixth Milestone: A* Path Finding

Estimated Time: To the end of project.
Assigned To: Matt Coley and David Ickert

- ✗ Initial A* path finding implementation.

1.1.7 Seventh Milestone: Enemy AI & Combat

Estimated Time: One to two weeks.
Assigned To: Matt Coley

- ✗ Enemies navigate towards player.
- ✗ Attack the player.
- ✗ Refine A* as needed.

1.1.8 Eighth Milestone: Implement User Interface

Estimated Time: 3 days to one week.
Assigned To: David Ickert

- ✗ Create UI graphics.
- ✗ Manage logic behind UI.
- ✗ Proper positioning and display of UI elements.
- ✗ Statistics screen.

1.1.9 Ninth Milestone: Polish & Code Refinement

Estimated Time: As time permits.
Assigned To: Matt Coley and David Ickert

2 Challenges

2.1 Getting Started

One of the first challenges we faced was getting started. Consequently, we met an additional time to discuss how we should proceed and to examine some questions that arose since the project proposal. One topic of discussion was how to handle the map. We decided, based on its uniqueness, to create it as its own class with its own draw and update methods instead of using the Sprite classes. We also discussed the size of the map and concluded that 2400 x 1800 pixels are sufficient. However, this is arbitrary and subject to change. Based on the types of objects in the game, we decided on the best draw order, game compents order, and discussed several possible game states.

Draw Order (back to front)

1. Map
2. Powerups
3. Nuts
4. Player
5. Enemies
6. Obstacles
7. Hometree
8. Menu

Game Component Order

1. Map

2. SpriteManager
3. Menu

Possible Game States

1. Active - When the game is being played.
2. Paused - A menu is displayed in the game.
3. Main_Menu
4. Game_Over
5. New_Round - Generate a new map.

2.2 The Sprite Class (David Ickert)

We quickly realized that the game engine did not need the original planned base class, GameObject. This class, from which all other classes derived from, had one attribute for position and two methods for the size. Since our game is 2D, and everything is composed of sprites, we decided to change the base class to an abstract Sprite class. This has the same capabilities as the GameObject, but adds textures, and extends to cover most other classes from static sprites to animated characters. More important, we incorperated two new virtual methods that allow polymorphism for both update and draw. Each sprite in the game is able update and draw itself as a result. This removes large amounts of code from the base update and adds a new layer of abstraction.

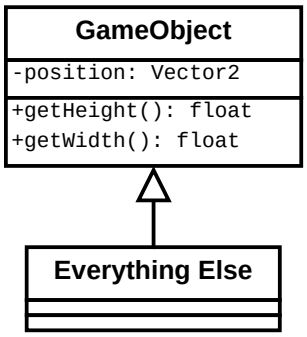

Figure 1: Abandoned Original GameObject Class

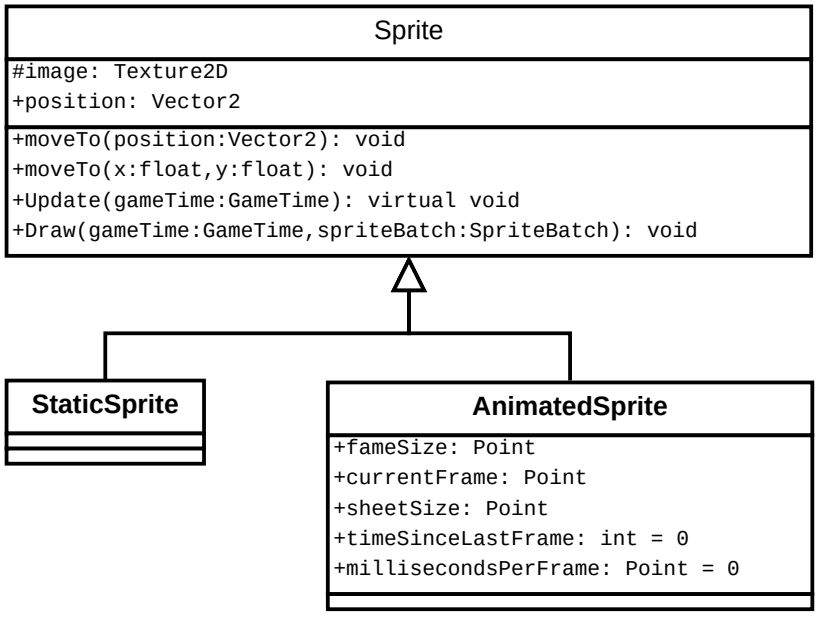

Figure 2: New Sprite Class

Creating the AnimatedSprite class was not too difficult. The most challenging aspect is the how to make changes during the update method to show the correct sequence. The first few attempts either did not animate correctly or failed to animate entirely. I expect that this class will need to be extended to handle multiple animations per a sprite sheet. For example, a character needs two unique animations when walking north versus walking south. For now, I used a simple sprite sheet I found online for testing.

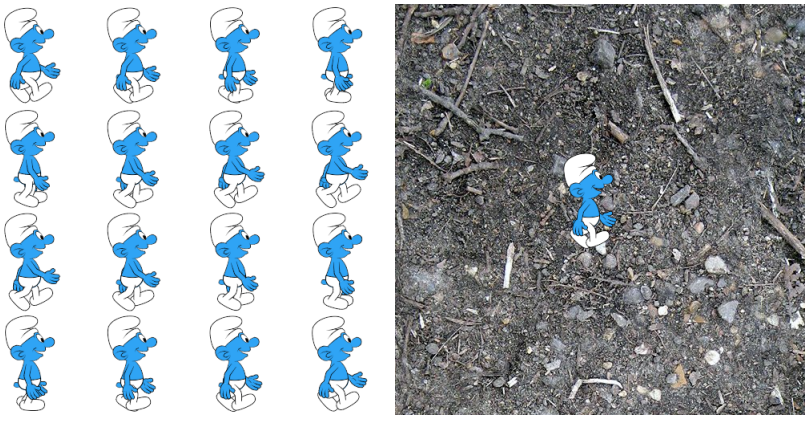

Figure 3: Left: Sample Sprite Sheet, Right: Animated on Matt's Map

2.3 The SpriteManager (David Ickert)

Our changes from the GameObject base class to the Sprite base class lead to the development of the SpriteManager class. This is an XNA game component and it has several important functions in the game engine. First, it overrides both the update and draw method to call each sprites respective methods. Second, it ensures that the game draws each sprite in the correct order previously discussed. Finally, it helps by adding that additional layer of abstraction and removes more code from the base update and draw methods.

2.4 The Map (Matt Coley)

When getting ready to implement the map, we realized that the map will end up being quite large, when measured in pixels. This meant our initial idea of creating a solid background texture would have been too large and hindered performance too much. We solved this problem by creating a tiled background texture that can repeat a ground pattern which is restricted to our texture size limits. The map also needed the ability to pan. This proved a larger task than we initially anticipated. We did not want to over-complicate the logic by introducing a camera, so we decided to make the map pan and leave the player in the same spot relative to the screen boundaries. This also meant that every item on the map needed to be updated whenever the player moved. We decided that the time saved and complications avoided with camera movement where worth the performance hit of the extra position updates each frame.

2.5 Game Menu (Matt Coley)

The game menu was fairly simple to set up. We decided to keep the menu drawing separate from the sprite manager. This allowed us to work simultaneously on both parts. Also, the rendering requirements for the menu system were very slim. We have basic pause, continue, and quit game functionality in the game menu.

2.6 Main Menu (Matt Coley)