SQUILL SQUISHER

Developed by Team 203

Paul Orvis - pworvis@wpi.edu, Thomas Meehan - tmeehan@wpi.edu

Final Design Doc

Genre:

Squill Squisher is a relatively simple action-based game primarily catering to casual audiences. It would likely be targeted towards mobile devices.

Game Description:

Players bounce a singular “bullet” projectile around the arena in an attempt to destroy all of the squishable objects (enemy aliens known as the “Squill”) in the arena. As the game progresses, the projectile moves faster and the Squills are harder to squish. The player is also able to unlock power-ups to supplement the bullet in their Squill squishing efforts. After the player loses all of his/her lives by letting the bullet go off screen too many times, it’s game over. Multiple difficulties are currently present, spread across 3 subsequent levels, affecting the above mentioned attributes. The “narrative” premise is that aliens are invading and you have to use stolen Alien technology to repel the invasion by bouncing a single, incredibly powerful projectile around the arena to defeat the Squills within.

Technical Features:

Some of the most advanced technical features of the game are having the projectile “bounce” off of surfaces in a fairly realistic manner. One of the hardest tricks to get right was seeing which part of an enemy the bullet would hit (left, right, top, bottom, corners, etc) and have it reflect accordingly. One of the most technically interesting features in the game is the way trig ratios were used to alter the angle of the projectile while preserving the absolute speed at all times. In addition, Squills have different numbers of hits that they can take before being defeated (this number will increase as players move through levels and enemies become tougher). The paddle that the projectile has to bounce off of on the bottom of the screen needed to move smoothly across the screen (i.e. holding the button to move it would make it move continuously instead of in discrete jumps).

Artistic Assets:

Not much was needed in terms of visual art assets. Most sprites were small and only a couple of characters in size. In fact, the only large sprites were things like the ASCII art titles. Powerup explosion sprites and other cool special effect-esque sprites would probably prove the most complex and demanding in terms of frames. Enemies in the arena needed to have multiple frames in their sprites to indicate damage as they are hit multiple times and begin to “crumble”. In terms of sound design, we ended up having about 10 sound effects and 9 sets of background music (though only 4 or 5 background tracks ended up being used in the final game). Sound effects were mainly played on events, such as when the bullet bounced or squished an enemy, when a power up was unlocked, or when a level was completed. Background music is unique to each level.

Implementation Plan:

In order to make the projectile “bounce” off the wall, we had to change how the object moves when it goes out of bounds or touches other objects. For example, if the projectile hit a wall 30° below the horizontal, it should move off the wall at 30° above the horizontal. We accomplished this by having 4 different functions that altered the direction based on the orientation of the surface it impacts. Changing the x and y velocities based on its new direction was handled by a custom function, effectively using trig ratios to change the components of the velocity without altering its absolute speed. To implement having Squills that take different numbers of hits to destroy, a counter was incremented each time that Squill is hit. Play(), pause(), and stop() were used to control the music throughout the game. In order to get the paddle to move continuously, we had to make the paddle move for every StepEvent. We used Prof. Claypool’s game engine to make our game, as both of ours were not working properly during development. All music is courtesy of Square Enix’s “Final Fantasy CHIPS” album, and all effects are pulled directly from Final Fantasy VII, VIII and I.

Distribution of Work:

In terms of distributing work, Paul handled the sprite design, and TJ handled the sound aspects of the game. All coding was more or less worked on as a team but in general more utility / systems related operations (such as handling the reflection of the ball at different angles) were handled by TJ while game design tasks (determining how many enemies go on screen, how many hits they’ll take, how fast the objects move) was handled by Paul.

Schedule:

We basically had one level fully playable by Thursday for the Alpha. This required we have sprite animation and object movement (i.e. projectile bouncing and paddle) done by Sunday, and more advanced gameplay such as powerups done by Tuesday. The only big shortcoming with the alpha was that powerups had to be called separately from main gameplay, as the linking wasn’t quite done. Sound tech could have also been better developed at that point. After the Alpha was done, we worked on integrating powerups, sound, and having multiple levels working properly on top of tweaking the gameplay so that it was ready Tuesday for the Final.