Qu-Dash
John Lee, Marcus Chiu, Andrew Halcomb

Game Back Story

The game starts with a Cube, named Qu, getting rejected by a sphere, who prefers a rectangular prism (skinny cube). The Cube then tries to make himself skinnier, but nothing seems to work. However, he accidentally tries too hard and ends up in the 2D world. This is the story of his journey to regain his 3D self… and rescue his princess from the actually evil rectangle.

Gameplay

Design

Qu-Dash will be a 2D platformer with some unique features including a movement cancel and time slow downs (details in Controls section). The 2 of the 3 levels we plan to design will try to emphasize each of those features individually, while the third will emphasize using them together.

The levels will also feature 3 types of enemies, one of which will be immune to time slow down, and another can pass through obstacles. The player will die in one hit, either from enemies or obstacles, but there will be several checkpoints on every level (specifically between highly technical areas) which may also serve as subtle hazard warnings. Enemies will also despawn after killing the player, naturally lowering the difficulty if the player is struggling, however, a point multiplier will be given at end based on the remaining enemies.

Controls

Qu-Dash will use simple WASD controls, with the addition of a high speed dash that's activated by double tapping one of the directional keys. We'll be replacing the conventional "spacebar to jump" with W, upward movement, and instead make 'spacebar' a movement cancel key. In other words, it's a way to adjust the dash distance as well as break falls and cut verticals by instantly setting the character's velocity to 0. We'll also be introducing a time slow down function using the 'shift' key which will slow down the entire game (player, enemies that aren't immune, and moving terrain). We hope that this feature will allow us to create challenging levels, without limiting the audience to "try-hards." To compensate, each level will be timed, and the timer won't be affected by time slow down. Whether this timer will be displayed on screen, or simply reported at the end of the level is TBD. We may just make it an optional feature.

Milestones/Schedule

16- October: Gather building blocks and main sprites
23- October: Implement movements and time slow downs.
30- October: Working player (combine art, code, animation)
6- November: Implement enemies, AI
13- November: Finish level 1
20- November: Working Prototype- All basic coding, 1 Level completed
27- November: Level 2 and 3 completed
30- November: Boss level/Full prototype
Due Date: polish levels, sprites, etc…

Accomplishment

A highly technical game, that is still playable at a casual level. A balance of difficult gameplay, handicaps, and rewards for not using them.

Work Breakdown

Art:

Main Character - Marcus, 1 week
Background Art - Team, 2 weeks
Element Art- Marcus, 1 week
Enemy Art - Andrew, 2 weeks

Systems:

Main Character movement- Andrew, 2 weeks
Enemy AI - Marcus, 2 weeks

- Base AI- 3 days
- Time Immune- 3 days
- Obstacle Immune- 5 days
- level specific- 3 days

Element Interaction - Marcus, 3 weeks

- spikes, bounce pads, moving platforms, falling hazards, checkpoint
- Each element- 4 days

Game State Machine - Andrew and John, 2 weeks
Boss AI - John, 1 week
Level Design- John, 3 weeks

UML class diagrams

Checkpoint
-posX: float -posY: float -active: bool = false
+getXPos(): float +getYPos(): float +activate() +deactivate()

BaseEnemy
-velocity: double -posX: float -posY: float +alive: bool
+kill() +setVelocity(double) +getXPos(): float +getYPos(): float

Player
-velocity: double -posX: float -posY: float +alive: bool
+respawn(float, float) +setVelocity(double) +getXPos(): float +getYPos(): float