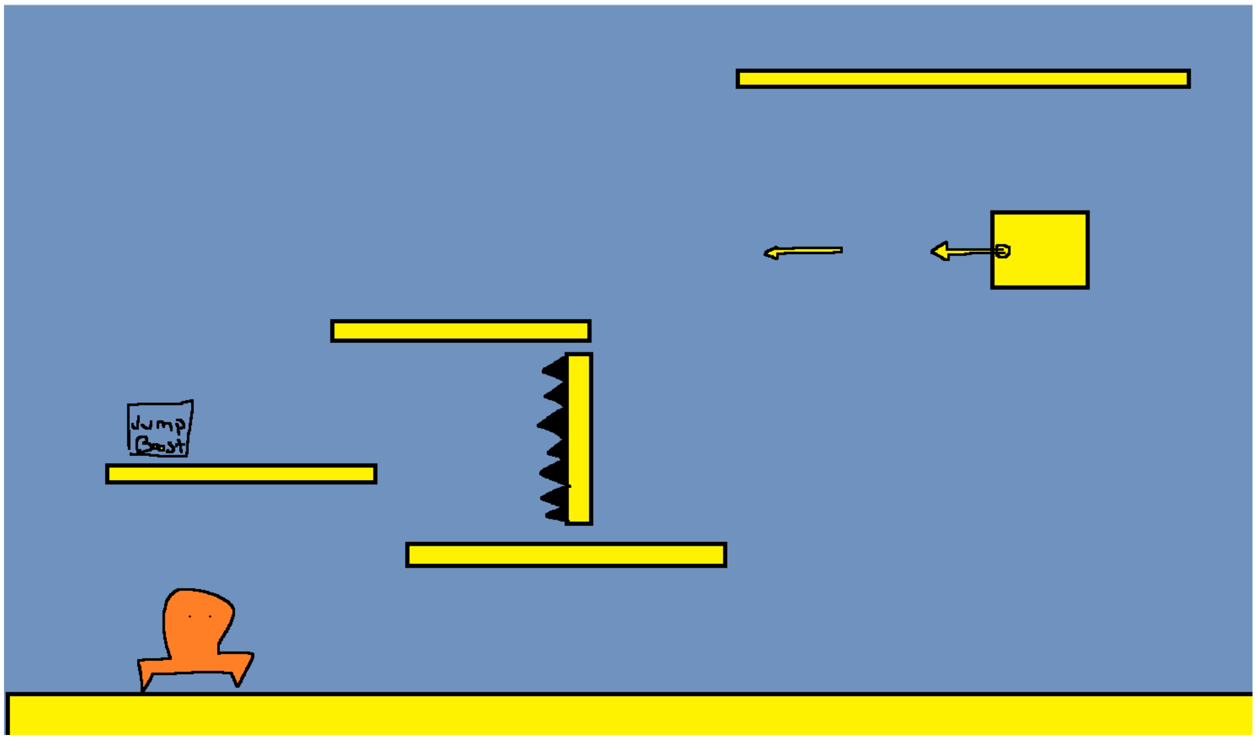
Marcus Macias
Don Tran
Maddux Rhodes

**Link to Github:** https://github.com/DonNamTran/Squid-Knight

## Synopsis

The proposed game will take heavy inspiration from Jump King as a platformer wherein the player controls a character that can jump on top of platforms and collect power-ups in order to get closer to the end of the game. There are obstacles along the way that can kill the character and if the character loses all of their lives, then the character is reset to the start of that current level. In addition to this, there are power ups in the game that would make passing a particular "Zone" easier for the player.

## Part 1 - Game Design



The game is about a squid knight on a mission to retrieve his stolen tentacle from a Captain Baha. The game consists of 10 levels, each with 4 zones that the player must navigate through in order to reach the top and touch a checkpoint to save their progress. You will be able to go back to previous levels or simply take a break then come back to where you left off. Each level introduces new platform types that require precise navigation so as to not accidentally hit a platform and fall back down to previous zones. Along the way, the player will find power-ups and flags that can help them progress faster or provide safeguards against missteps.

The game will have gimmicks in the form of power-ups and flags that the player can acquire. The player will be able to use power-ups like jump boost, phase through platforms, extra life, etc. These will help them navigate difficult areas that are hard to navigate without them.

Flags will allow the player to save their progress and pick up where they left off or replay previous levels.

The game controls will allow the player to jump a certain distance based on how long they hold down the spacebar. In the air, the player can strafe left or right using the A or D keys. The game is designed to require precision and careful navigation, with new platform types introduced in each level. There will also be obstacles other than platforms like spikes or small machines shooting arrows in a direction.

**Part 2 - Development Design**

Our platforming game will follow the model-view-controller design archetype. The model class will contain our different game objects; specifically, the player character, the various enemies (potentially bosses?), obstacles that could kill/damage the player, the background, the level's platforms, and power up sprites. Each of our levels will most likely be separate modules (files) that will be loaded whenever the player selects that level. The model will keep track of the player's position, the enemy positions, and power ups that are spawning. Whenever a player picks up a power up, the model will update the player's attribute accordingly, and then remove it whenever the power up duration runs out. Whenever the player loses a life, the model will need to be updated back to the last saved checkpoint. If the model detects the player has lost all their lives, then it will reset the player back to the beginning of the level.

The controller layer will move the player based on the input provided. The amount of movement will depend on the power ups affecting the player. The player's main movement to navigate the level will be jumping. Jumping can be done with both space and the up-arrow. The player will also be able to use A/D or left-arrow/right-arrow to strafe left or right in the air and on platforms to dodge enemies. That is all the movement that the player will be able to dictate. The controller layer will also pass relevant information to the model layer. Like if it detected that the player collided into an enemy or power up, or if the player moved off the screen (the scene will change to the next part of the level). The view layer will take the objects from the model layer and display it on the screen. It should be updated frequently to reflect the constant updates that will happen in the model layer.

The UI will be pretty simple. It'll start off with a start-button and then a level-select. During the actual level, the player will have a lives meter, potentially a progress bar that shows how far the player is in the current level, and that will most likely be the extent of the player's UI. Once the player completes the level, they'll be sent back to the level select menu where they can continue on to the next level.

There are a couple of potential issues that we can see happening. First is how we will have the controller layer communicating with the model layer to update what's happening to the player. The next is refreshing the game back to a checkpoint whenever a player dies. There might be performance issues depending on how many enemies we will have present on the screen, as well as how we deal with the player frequently moving between screens as they go up and down the level.

**Part 3 - Division of Labor**
Marcus - Gameplay focused, player mechanics (5 hrs), particle systems (5 hrs)
Don - "Gimmick" focused and user control (5 hrs), art assets (5-10 hrs)
Maddux - Level design and level mechanics focused (5 hrs), checkpoint system (3 hrs), art assets (5-10 hrs)

Conflicts or Issues: We went over potential issues in the previous part.

1. Controller layer communicating with the model layer
    a. Fix: Have well defined classes from the start, and have each class communicating with the ones that are supposed to communicate.
2. Refreshing the game back to a checkpoint:
    a. Fix: Store the player's previous progress whenever the checkpoint is hit and if the player dies, reset progress to that.
3. Performance Issues (a lot of enemies/sprites):
    a. Fix: we could limit the amount of entities allowed on the screen at one time.

**Part 4 - Timeline**
Milestone 1: March 30

- Marcus - Implement the "jump and fall" mechanic, make sure that the gravity is working, etc.
- Don - Implement the basic functionality of the power ups, make sure they work properly. basic rendering
- Maddux - level scroll when the player gets to the top of the current level, Implement platform physics where the player collides with those, making sure you can't go through the bottom, etc.

Milestone 2: April 12

- Marcus - Implement the "stronger jump" mechanics, where the player holds the jump button and you can jump higher based on how long you held the button.
- Don - Finish power ups and implement full user control
- Maddux - implement checkpoint system with certain levels for saving progress.

Milestone 3: April 26

- Marcus - Implement particle systems
- Don and Maddux - Finalize art assets

Marcus Macias
Don Tran
Maddux Rhodes

Final Game Submission: April 26
Final Exam Presentation: May 4

***MILESTONE 1 CHANGES/UPDATES BELOW***

## Part 1 - Updated Game Document

We changed the "lore"of the story because a dragonslayer stealing the tentacle of a squid didn't make any sense. Since we started working on the project, we went from a little grey block that couldn't jump or move at all, to an entire level design (really basic but we're getting there) and a sprite of an actual squid that can run and jump around and collect power ups and increase jump height with that power up (we haven't implemented all of the power ups yet, we postponed that functionality).

Updated Project Timeline is listed below in Part 2.

Postponed Tasks:

- All power ups being functional has been postponed until milestone 2.
- The sprite for the player has been moved up to being complete (the power up sprite and player sprite while having a power up have yet to be implemented)
- Level scrolling

Technical Challenges are listed below in Part 3.

The final game design will not change.

## Part 2 - Updated Project Timeline

Milestone 1: March 30

- Marcus - Implement the "jump and fall" mechanic, make sure that the gravity is working, etc. (Marcus 90%, Don 5%, Maddux 5%)
- Don - Implement the basic functionality of some of the power ups, make sure they work properly. basic rendering (Don 95%, Marcus 5%)
- Maddux - Implement platform physics where the player collides with those, making sure you can't go through the bottom, etc. (Maddux 60%, Marcus 40%) Player Sprite (Maddux 85%, Marcus 10%, Don 5%)

Milestone 2: April 12

- Marcus - Implement the "stronger jump" mechanics, where the player holds the jump button and you can jump higher based on how long you held the button.
- Don - Finish the rest of the power ups and implement full user control

- Maddux - Implement the level scrolling, implement checkpoint system with certain levels for saving progress.

Milestone 3: April 26

- Marcus - Implement particle systems
- Don and Maddux - Finalize art assets

Final Game Submission: April 26
Final Exam Presentation: May 4

**Part 3 - Technical Challenges**

- The team encountered a technical challenge with the collisions where the player jus teleports to the top of the ground above
- There was also a problem where the power ups stack and it could go through the terrain

These challenges affected the development timeline by having to postpone some of the powerups until a later milestone. The collision and teleport glitch was solved.