TP3 UPDATES:

* The ‘antagonist’ is now a bomb that appears at random intervals on the screen
* I have implemented a coin system for scorekeeping

TP0: Project Proposal—*Leap*

Julia Liu

1. **Project Description: *Leap***

In this proposal, I will outline the basic steps that I will take to complete my project. The project’s working title will be *Leap,* and it will involve a game where the user attempts to jump from block to block to get as high as they can. As they jump higher and higher, the character will face various challenges. For example, the blocks may get smaller, and there will be a random occurrence of “special blocks,” which are blocks with special functionalities such as disappearing blocks, tilted blocks, or blocks that have spikes on them that will kill the player. The objective of the game will be to collect coins from time to time. There will also be a malicious bird that appears from time to time, stealing coins that the player collects. This bird will only be allowed to move from the top of the screen to the bottom. The player can escape from the bird by jumping to a different block above. The player dies if they miss jumping on a block and fall all the way down.

1. **Similarities from past work and differences**

Past work, like *DoodleJump*[[1]](#footnote-1), features a character that jumps from block to block with no apparent motive in mind other than to keep jumping. I will gamify this concept to become more engaging by incorporating reward systems, creating an antagonist, and introducing more complexity (different types of blocks).

Moreover, past popular games like *NinJump*[[2]](#footnote-2), while including a different range of actions, does not include the same range of threats that I will allow my character to face. Mostly, games that involve survival and traveling farther distances typically involve more avoidance of static threats. My game will involve more active avoidance of dynamic antagonists, like the bird character that I plan on incorporating.

1. **Structural Plan**
2. **Sprite class**

The sprite class, as the name suggests, will handle the player’s character. It will include functions that dictate where and how the player will be allowed to move on the screen, the status (dead/alive) of the player, and the player’s stats, such as their current score, current health level, current number of coins collected, and other related information. The sprite gets controlled by left and right arrow keys.

1. **Block creation class**

The block class will handle the creation and randomization of placing the next block and loading it offscreen. The block creation class is also responsible for randomizing the occurrence of special blocks, like disappearing blocks, tilted blocks or other blocks that can kill the player.

The block class updates the player's status based on whether the player experienced a block-related death.

1. **Antagonist class**

The antagonist (most likely to be a bird) class will occur at random intervals throughout the game and attempt to kill the player based on the player’s location. If the antagonist makes a collision with the player, the antagonist will steal the player’s coins and fly away.

1. **Visuals and UX class**

This will create the background, check for events that require sounds, such as leaps, encountering deaths, and ambient background music for the menu, intense music for in-game moments, and sad music for the player’s death.

1. **Menu Screen and death screen displaying class**

Will check on the output from the sprite class and determine if the death screen should be displayed. Will check on other mouse inputs to determine when the menu screen should disappear and whether the game started is true.

1. **Algorithmic Plan**
2. **Antagonist Class**
3. **Collision animation**

I plan on calculating how many coins will “explode” when the antagonist crashes into the player. I will calculate the magnitude of the explosion based on how many coins are currently in the player’s sack.

1. **Player tracking (with Dijkstra’s graph algorithm)**

I plan on using Dijkstra’s graph algorithm to track the player and locate them. I must ask a TA for help on the specifics of this! I read the documents online and I must ask some questions to clarify.

1. **Block Creation Class**
2. **Orthogonality of character on the tilted block + b. Jump physics**

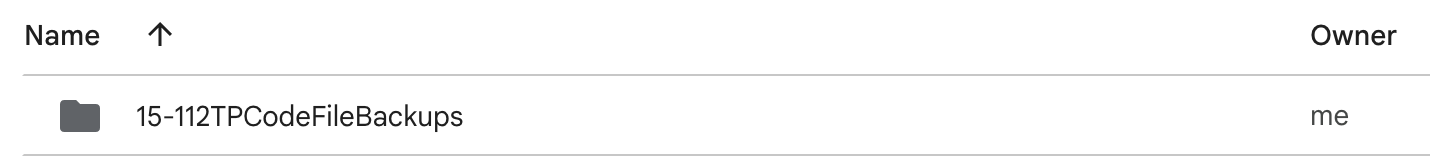
The tilted blocks will all be tilted at a specific angle, (perhaps 30 degrees). This will impact the direction that the character jumps in. Additionally, I will always code the character so that it is orthogonal to the block.

1. **Timeline Plan**
2. **Pseudocode completion**

I will be aiming to complete the pseudocode outlining by April 11th

1. **Block class (basic) and sprite functionality.**

The sprite and the block class should be completed by April 13th.

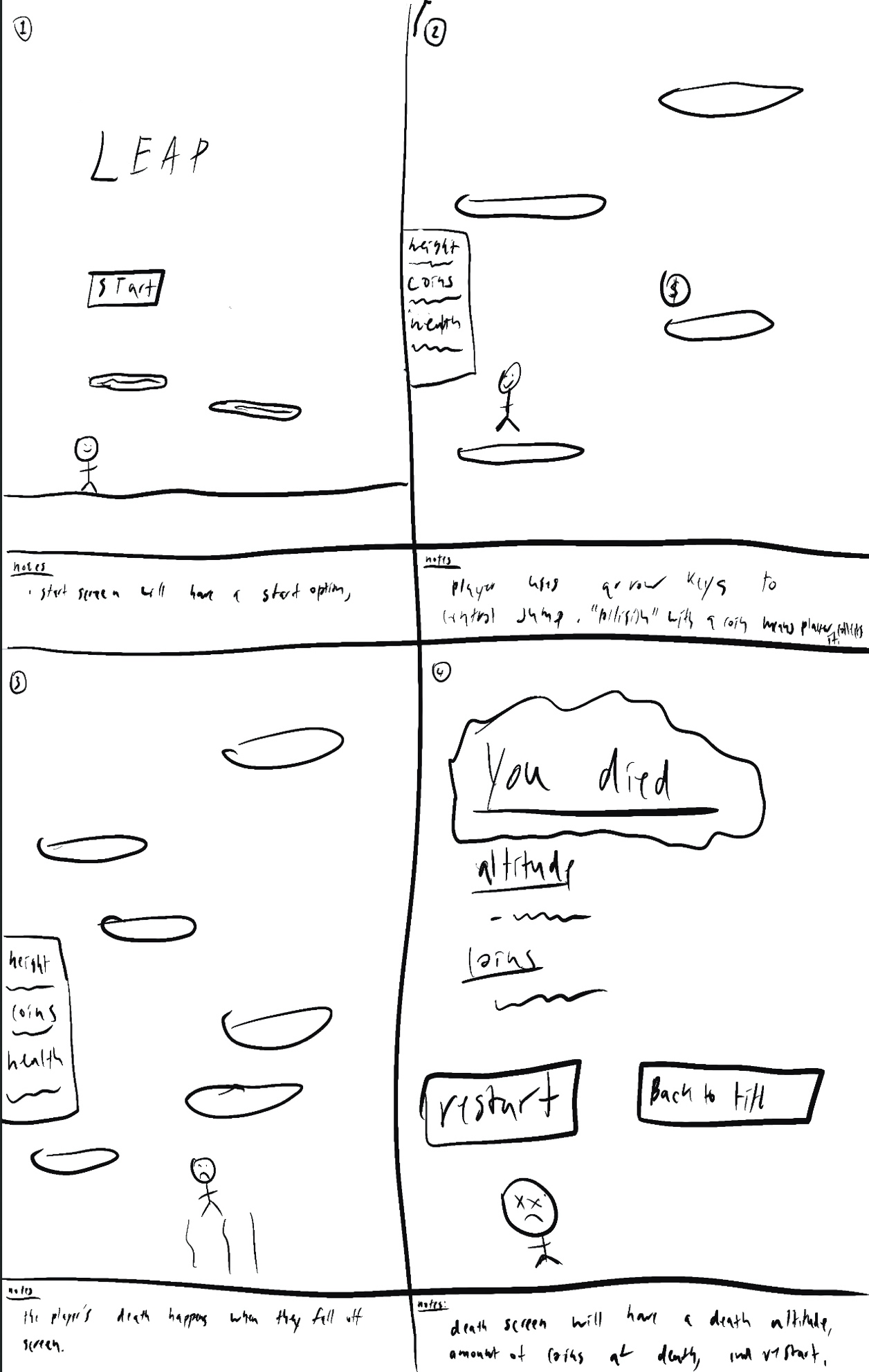
1. **The antagonist class and block class in its full complexity should be completed by April 17th.**
2. **UI/UX features should be implemented by April 18th.**
3. **The remaining time should be used to tweak details and debug and search for edge cases.**
4. **Version Control Plan**

After every two hours of working on the project, I plan to upload a code file into Google Drive that will be named according to the format of “date\_time\_majorChanges.”

1. **Module List**

I do not plan on including any external modules. I will be using CMU graphics and the modules that we use in class.

1. **Storyboard**



1. Panel 1

The start title screen will feature a start button that the player will press a key to start. Instructions on exactly how to start the game, i.e. which key to press, mouse event, etc. yet to be decided, but will be displayed on the start screen.

1. Panel 2

While in the game, the player will be able to jump from block to block.

1. Panel 3

The death event occurs when the player misses a block and falls off the screen.

1. Panel 4

Death screen will occur after the character drops off the screen. It will feature sad music, report the altitude of the current game, and also the number of coins collected. There will be a restart option and a back-to-title screen option.

1. Panel 5

Shows the antagonist in action, appearing at the top of the screen. Also featured in this panel are the “special” blocks, i.e. the tilted blocks.

1. Panel 6

A collision event between the antagonist and the player creates a small burst of coins based on how many coins that player currently has. (note, the antagonist will only start appearing if the player’s current coin count is non zero.

A picture containing diagram

Description automatically generated

Note: no differences to this document has been made for TP2

1. https://doodlejump.io/ [↑](#footnote-ref-1)
2. https://apps.apple.com/us/app/ninjump-run-jump-game/id1535414420 [↑](#footnote-ref-2)