Game Proposal: TIMELOCK

CPSC 427 – Video Game Programming

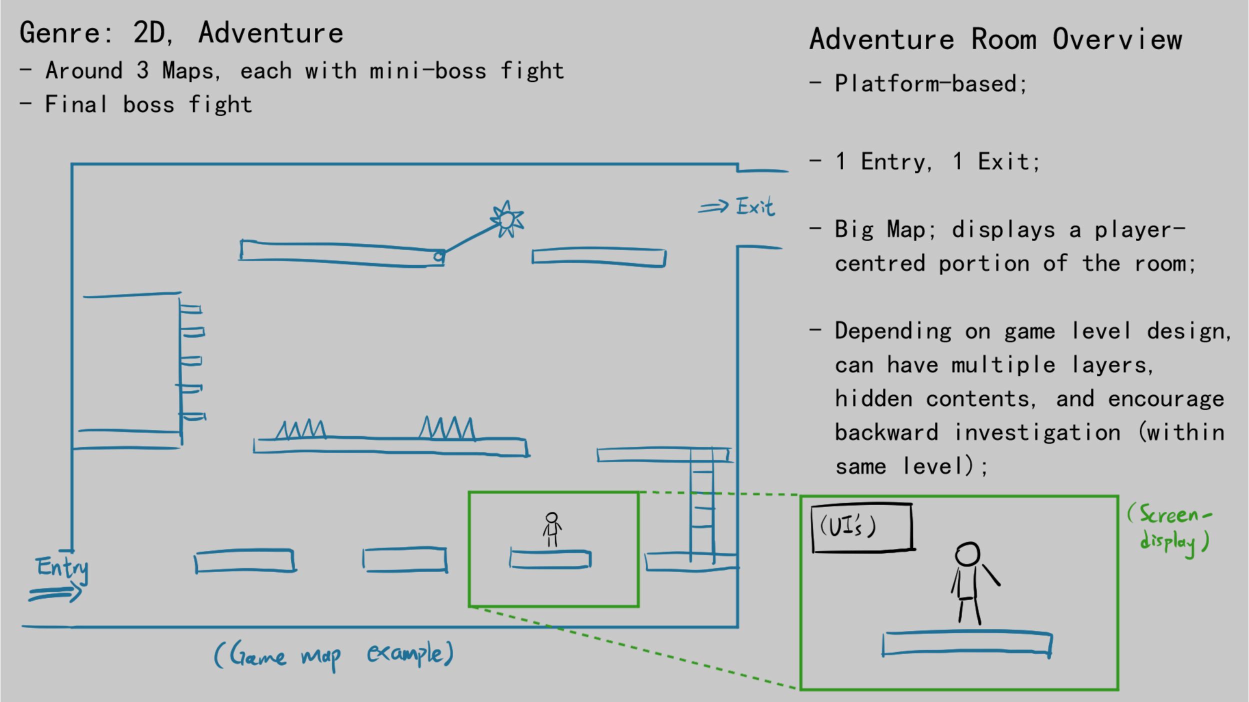
Team: Group 03

Julia Rees 80236615
Dieter Frehlich 14848410
Yixian Cheng 94548492
Kevin Zhou 35883164
Matias Gauvin 27796267

Story:

The game will take place entirely in a Clockwork Castle, which the player wakes up to find themselves trapped in. To escape the castle, the player will need to successfully traverse through 3 rooms, each complete with their own mini-boss, followed by a final boss. Each level, and its mini-boss level, will correspond to a single time manipulation mechanic (decelerate, accelerate, rewind) in order to help the player build their skill in the mechanic. The final boss level will require the player to use all three mechanics in various successions in order to succeed.

Scenes:



Genre: 2D, Adventure

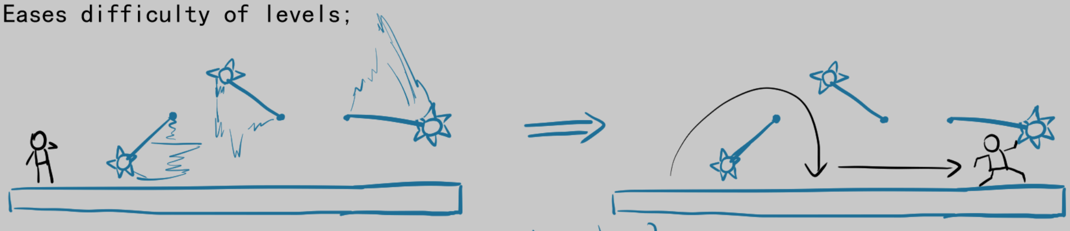
- Around 3 Maps, each with mini-boss fight
- Final boss fight

Adventure Room Overview

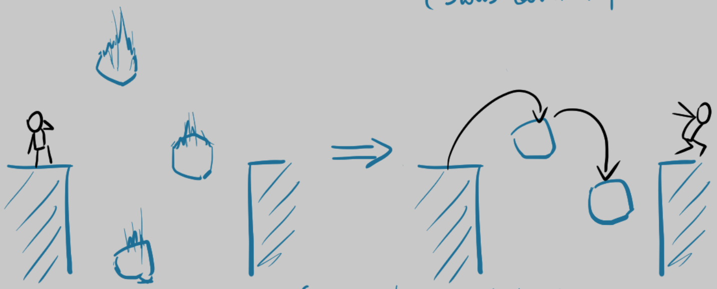
- Platform-based;
- 1 Entry, 1 Exit;
- Big Map; displays a player-centred portion of the room;
- Depending on game level design, can have multiple layers, hidden contents, and encourage backward investigation (within same level);

Time Decelerate

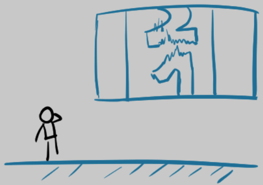
- Slows down motions of the environment;
- Allows interaction with fast-moving objects;
- Eases difficulty of levels;



(Slows down traps)



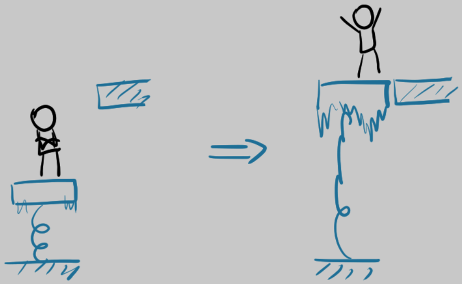
(Turns falling rocks to platforms)



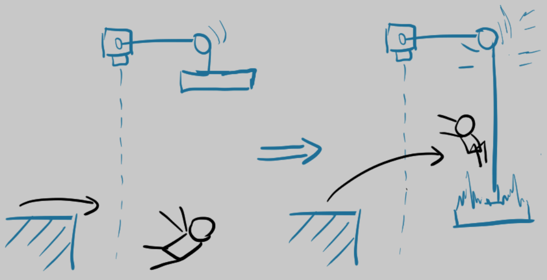
(Read fast-scrolling passwords)

Time Accelerate

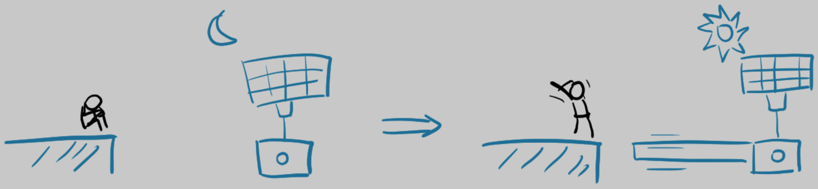
- Makes certain objects faster (e.g., supports & propulsion);
- Overloads hostile objects;
- Switches daytime;



(Larger Propulsive Force for faster objects)



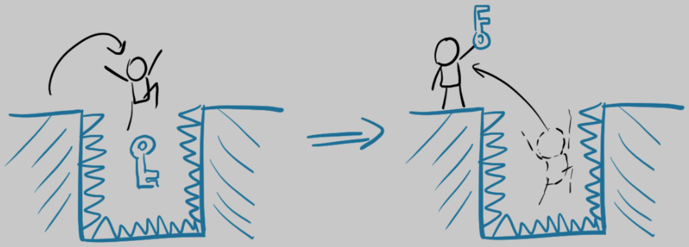
(Platform released quicker upon detecting Player)



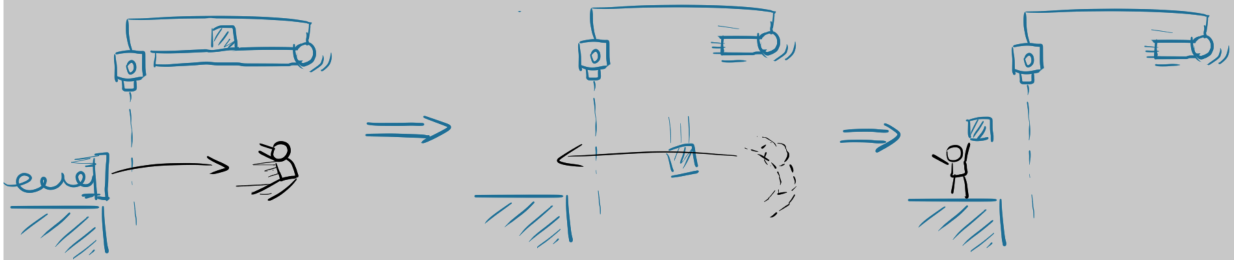
(Difference in terrain for Day vs. Night)

Time Rewind (Applied to Player)

- Shifts back to a previous position -> Allows undoing actions;



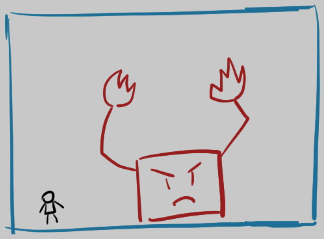
(Undo one-way trip)



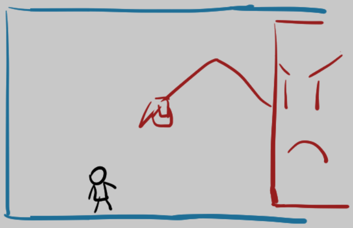
(Retrieve Falling object)

Boss Fights

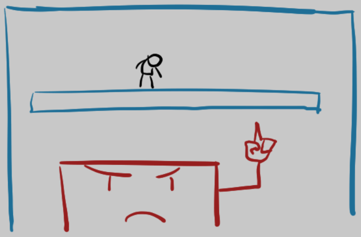
- Depending on boss fights, the room & boss location can vary, but will likely have a fixed camera displaying everything in the scene;



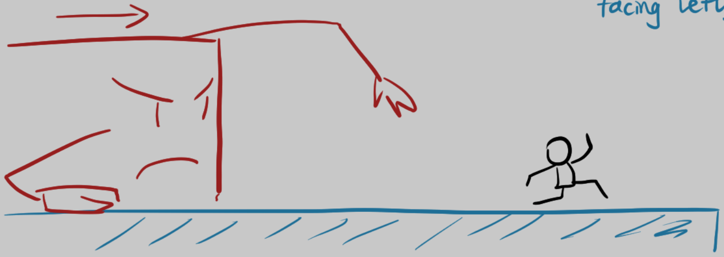
(Type 1: Boss at the center)



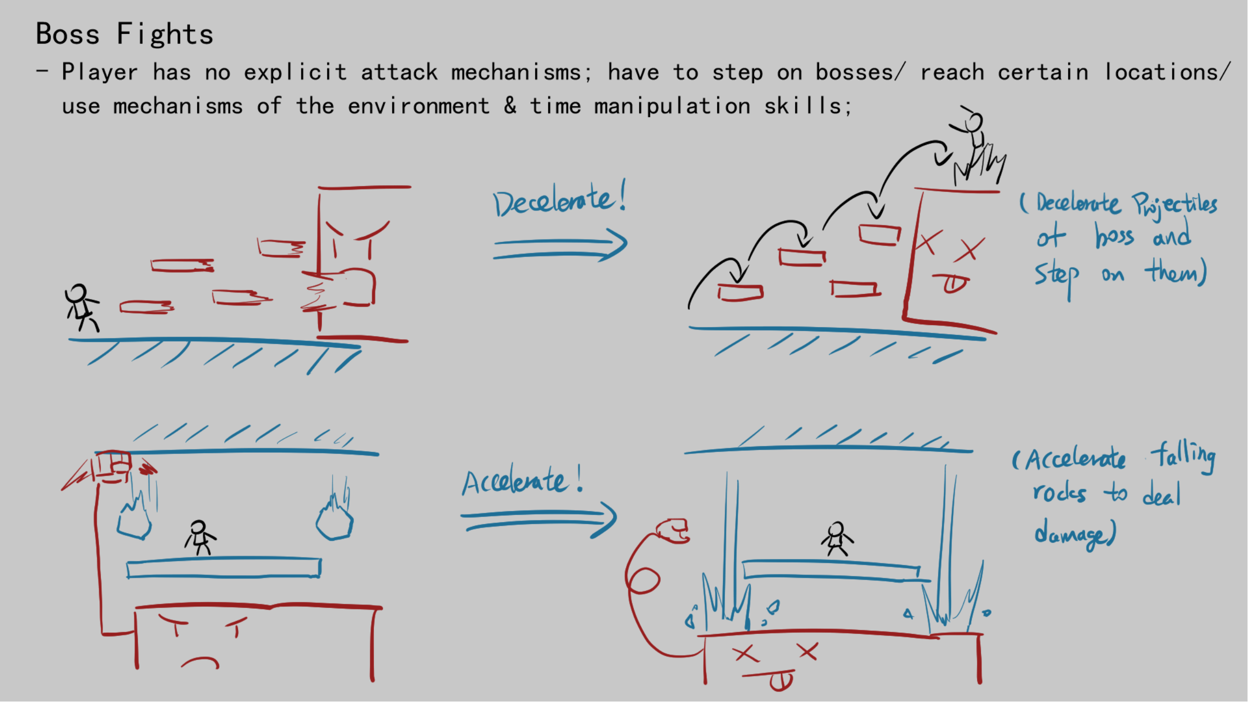
(Type 2: Boss on the right, facing left)



(Type 3: Boss at the bottom)



(Potential Chase Scenes)

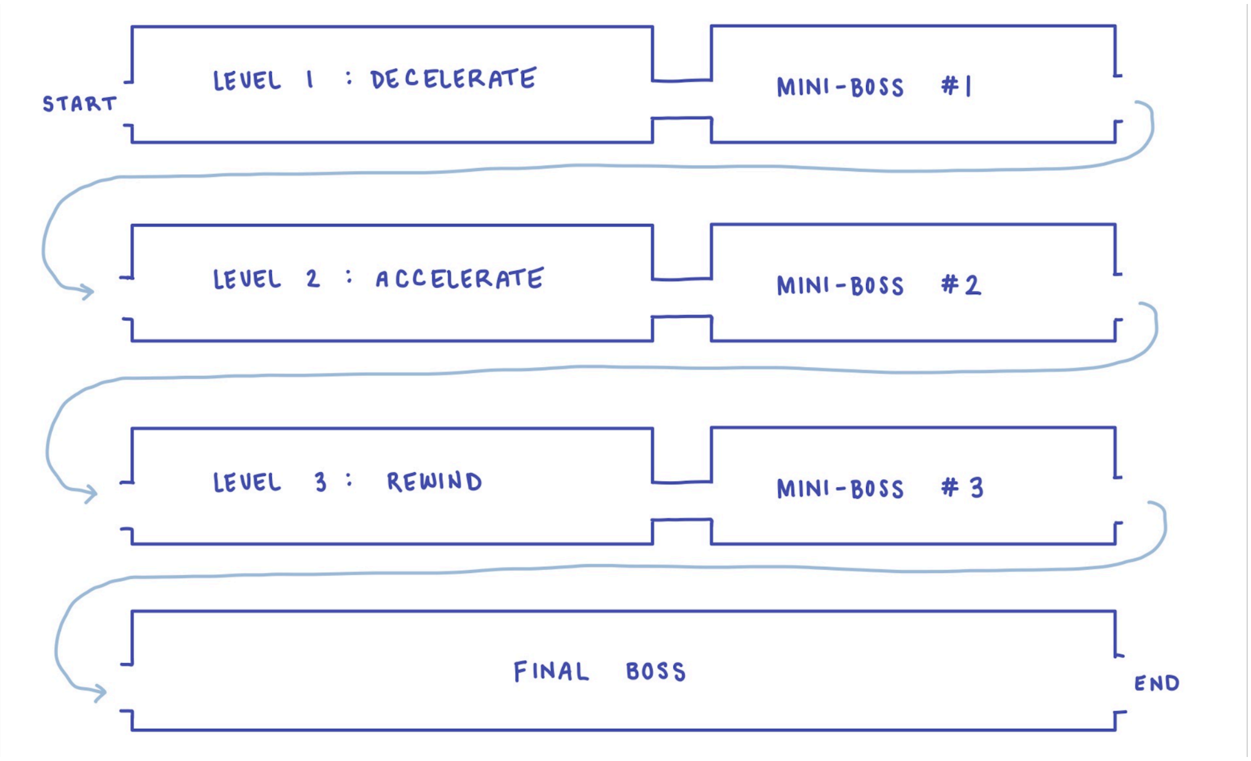


Boss Fights

- Player has no explicit attack mechanisms; have to step on bosses/ reach certain locations/ use mechanisms of the environment & time manipulation skills;

Decelerate!
(Decelerate Projectiles of boss and Step on them)
Accelerate!
(Accelerate falling rocks to deal damage)

Map:



Technical Elements:

Game Mechanism - Time Decelerate: Time Decelerate will allow players to slow down motion in their surrounding environment, enabling interaction with fast-moving objects, dodging obstacles, or reading fast-moving text on the screen. This ability will likely have a short cooldown period before re-use.

Game Mechanism - Time Accelerate: Time Accelerate will move certain objects faster, increasing propulsion and momentum from platforms which may allow them to reach higher or farther areas. Like Time Decelerate, this ability will have a short cooldown period before re-use.

Core Technical Requirements (Effects):
Rendering:
Our game renders basic 2D elements in the foreground for the player to interact with, which will be supported with both art assets and shader scripts. For instance, the background moves slightly slower than the foreground elements to create a parallax effect. In terms of art and sound assets, we will use custom designs where possible, with royalty-free assets as a fallback option.

Assets:
We plan to design characters and foreground objects with elements related to time, such as pendulums, clock gears and hourglasses. Different background assets will be used for different levels.

Sound:
We plan to manipulate the speed and style of the background music in accordance with the player's manipulation of time. We will also include sound effects for jumping, time manipulation, and player collisions.

Core Technical Requirements (Engine):
2D:
The camera will have a side-view perspective, centering the player during normal gameplay. During a special scene such as a boss fight, the camera will pan out to show both the player and the boss. The player will collide with walls and platforms, and enemies. There will be some dynamic objects which move around, such as a floating platform or swinging clock pendulum.

Logic:
The player will be able to walk left/right and jump. The user also has the ability to speed up, slow down, or rewind time. This ability will have infinite use, but there will be a short cooldown period after.

Physics:
The player, enemies, and any movable object will be affected by gravity. There will also be a simple friction system in place so that the player slides off steeply angled surfaces.

AI:
Basic enemies will follow a predefined, repeating path, while the more complex bosses will follow the player, targeting them with attacks.

Advanced Technical Elements:

List the more advanced and additional technical elements you intend to include in the game prioritized on likelihood of inclusion. Describe the impact on the gameplay in the event of skipping each of the features and propose an alternative.

Game Mechanism - Time rewind: While we plan to implement all three time manipulation mechanics, incase of unforeseen circumstances Time Rewind will only be included if time permits. Since our game will still have two time manipulation mechanics we believe it is a strong game without this mechanic, and do not plan to have an alternative for Time Rewind but to instead lengthen the duration and complexity of the levels featuring the other two mechanics.

Rendering - Customized Shaders and Multi-pass Rendering: Customized shaders (mostly fragment shaders for 2D game) and multi-pass rendering could enable engaging visual effects. The visual feedback for certain events (e.g., ability activation, cutscenes) could be enhanced, and status of entities in the game can be made more intuitive (e.g., green outline for benign entities, red outline for hostile ones). Implementing customized shaders and multi-pass rendering introduces additional risks due to unconventional mathematical calculations and external assets; therefore, these features should be considered only if time permits.

Devices:

Explain which input devices you plan on supporting and how they map to in-game controls.

Our game will be targeted for laptop and desktop computers. It will be designed to run on Mac, Windows, and Linux.

We plan to natively support keyboard inputs for all in game controls, and potentially support controllers through external software that maps controller inputs to keyboard inputs. The keyboard controls will resemble the following:

1. A/D + left/right arrows for moving left/right
2. W/ Space/ up arrow for jumping
3. Other keys to activate time control powers
    a. Could use number keys (1, 2, 3)
    b. Alternatively, could allocate one button for “using” an ability and give the player the option to choose an “active” ability.

Ideally, we give users the option to use the control scheme that feels most natural to them.

Tools:

Specify and motivate the libraries and tools that you plan on using except for C/C++ and OpenGL.

Audio: To simplify loading and playing sounds and game audio, we will likely use an audio library.

- OpenAL is a good candidate. The library is primarily for 3D sound, but we can also use it for 2D games. The API is simple and similar to OpenGL.
- glText: library for rendering texts in OpenGL.
-

Team management:

Identify how you will assign and track tasks and describe the internal deadlines and policies you will use to meet the goals of each milestone.

The team as a whole will assess the work to be done in each milestone, and assign tasks based on preferences, but ensuring that workload is roughly spread evenly. Dedicated discord subchannels will be made specifically for the broad tasks and goals for each 2 week milestone. Each person will send messages into the respective chat to discuss progress and identify any issues or related information.

We will have an in person **weekly meeting on Wednesdays**, in tutorial if possible, and at **6:30pm** otherwise.

Work Distribution:

1. At the beginning of each milestone, we will add each weekly task to a Trello board.
2. Everyone individually will assign themselves to the task they want to work on the most
    a. If there are unassigned tasks, or too many people on a given task, as a team we will decide how to redistribute
3. In our weekly meetings we will do a rudimentary standup:
    a. Say what you are working on, what has been done, what is blocking you, and if there are any ways the rest of the team can help.

Development Plan:

Milestone 1: Skeletal Game DUE FEB 9TH

For milestone 1 [M1], you should have a basic version of your target game. It should incorporate basic rendering, input-driven response, 2D motion, correct basic collision handling, event-driven/random response, and a minimal set of assets

Week 1 (Base Engine) - **Jan 26 - Feb 2**:

1. Diagram out basic ideas for project implementation

In parallel:

1. Refactor existing ECS -- generalize to arbitrary components.
    a. Work from diagram of ECS design
2. Rendering system
    a. MVP: API to enable rendering for a scene -> provide list of entities with position components, render a dot for each position. TODO: more detailed requirements.
    b. Start implementing basic camera logic
3. Asset creation

Week 2 (Base Systems) **Feb 2 - Feb 9**:

1. Implement motion system
    a. Keyboard inputs move character around (left, right, jump)
    b. Some entities can follow a predefined path.
2. Implement collision system
    a. Two entities with collision components should not be able to occupy same position
    b. Potentially implement a few tiles/obstacles/traps/enemies to test collision testing & scene construction
3. Implement decelerate
    a. Any entity with ‘decelerate’ and position components should update slower when player triggers decelerate
        i. Global ‘scaling’ factor -> each time we update position we use the scaling factor to determine speed. When deceleration is triggered, this factor slows down.

Milestone 2: Minimal Playability Due March 2nd

For this milestone, you should continue to support **all required** skeletal game features and fix any game-stopping bugs (P0 and P1). You should augment those features with core gameplay logic, incorporate additional assets and play elements that allow for non-repetitive gameplay, fix all known bugs, and perform playability testing.

Week 1 - **Feb 9 - Feb 16**

1. Bug fixing if needed
    a. Perform some basic testing as a team. Note down any bugs we encounter.
2. Refactoring + buffer time
    a. If possible, generalize decelerate logic to work for *time accelerate*.
    b. Clean up any messy logic from M1
    c. Some extra time to implement anything missing from M1
3. Start level design + boss battle mechanics design (mini-boss + final boss)
    a. Come up with prototype designs

Week 2 (Reading Break) - **Feb 16 - Feb 23**

1. Continue level design
2. Start assets + design elements
    a. In preparation for level loading system, design and implement a few in-world objects/traps/enemies;
3. Implement Accelerate (if not already implemented in Week 1).
    a. Any entity with 'accelerate' and position components should update faster when player triggers accelerate
        i. Global 'scaling' factor -> each time we update position we use the scaling factor to determine speed. When deceleration is triggered, this factor slows down.
4. Start designing rewind-related gameplay elements
    a. Level-related
    b. Mini-boss

Week 3 - **Feb 23 - Mar 2**

1. Level loading system
    a. Allocate time for researching potential pre-existing solutions.
    b. Take scene descriptor file and generate level in game
    c. Translate prototype designs into scene descriptor files
2. Continue with Rewind implementation + design
    a. Focus on implementing the mechanic in the engine

Milestone 3: Playability Due Mar 23

For this milestone, you should have a completely playable game, continue to support all required features from prior milestones, and have fixed the majority of your latent bugs. In addition, you should now also include advanced features such as detailed geometry, non-linear motion, and time-stepping based physics. Your game should provide robust continuous play with no memory leaks, crashes or glitches.

Key to this milestone is that you test the playability of all new features and ensure alignment with your team's game development plan. You are encouraged to ask people outside your team to play test your game, so you can fine tune your gameplay and interface.

Week 1 - **Mar 2 - Mar 9**

1. Start looking online for sound effects
    a. Begin building library of relevant sound effects
2. Consult external sources for music design
    a. Post on reddit threads for anyone interested in producing music?
    b. Joaquin (Dieter's friend).
3. Intro/Menu logic
    a. Splash screen, level selector?, pause/settings menu, etc

Week 2 - **Mar 9 - Mar 16**

1. Story design
    a. Narratives and Dialogues
    b. How to present story to player
2. Group Gameplay Testing!!
    a. As a team, sit down and play the game. Talk about honest thoughts and decide on key areas for improvement.

Week 3 - **Mar 16 - Mar 23**

1. Finetune mechanics (accelerate, decelerate, rewind (if implemented))
2. Finalize gameplay contents (3 levels + bosses)
3. Sound integration
    a. Play sound effects after certain events -- ie. use of time mechanics, jump, collide, etc

Milestone 4: Final Games Due Apr 6th

The final version of your game should support robust and continuous gameplay and integrate advanced game elements. It should be self-contained, i.e., players should be able to play the game with no outside help or instructions other than those provided by the game itself. You should implement one or more additional advanced gameplay feature (see below) and you are welcome to incorporate advanced features using third-party libraries (verify with TA). Your final video will demonstrate your final game, highlight the advanced features, and provide a review of your game's development and evolution.

Key to this final milestone is that you test the playability of all new features and ensure alignment with your team's game development plan. You are encouraged to ask people outside your team to play test your game, so you can balance and fine tune your gameplay and interface.

Week 1 - **Mar 23 - Mar 30**

1. Tutorial Development
    a. Decide on how to explain the game and show the features to the user
2. External playtesting -- sanity checking how intuitive the game feels.
    a. Ask guided questions to gather specific smaller insights for fine-tuning
2. Time to implement suggestions + fine tune smaller elements of game "feel"

Week 2 - **Mar 30 - Apr 6**

1. Wrap up advanced visual effects with shader scripts and multi-pass rendering.
2. Finish refinement of camera functionalities (follow Player/ fixed position/ screen shaking/etc.)
3. Extra buffer time for anything missing!