

## Project Schedule

*Name: Beat Breaker*

Student Name: Jake Fitzgerald

Student No: C00288105

Phase 1 (November 10th - November 30th)

Phase 2 (December 1st - December 24th)

Phase 3 (January 1st - January 30th)

Phase 4 (February 1st - February 30th)

Phase 5 (March - April) [FINAL PHASE]

## Phase 1: Core Systems & Setup (Weeks 1 – 2)

- SFML Project setup with file structure.
- Input handling:
  - Input / Event handling type (Keyboard keys, Joystick controller)
  - Latency and grace period for detecting if the input was on beat.
- Basic scene layout (Menu, Options, Gameplay, Results, etc).
- Visuals
  - Window resolution (static/dynamic)
  - Sprite sizes, consistent numbers (64 by 64 pixels, etc)
  - Use debug SFML shapes for testing (toggle with key)
  - UI Buttons for scene navigation (clickable), Font import and basic text.

Simple character controller (physics, gravity, friction (air/grounded), collision detection)

- States: [Horizontal movement, Jumping, Falling, Block breaking]
- Simple animation
- Basic animations (move left/right,jump, falling, hit)

Character gameplay attributes:

- Health timer (slowly depletes based on BPM count of the current MIDI track).
- Lives (how many times the player can fail during a run).

Gameplay HUD

- Score Counters [Current Score, Personal Best]
- Health Gauge
- Lives Counter
- Track “Deepness” (How far you are into a song).

Pause Menu

- Pause/Retry when game is paused.
- Stop the timings for the MIDI track and be able to resume at the same place.

Main Menu

- Simple clickable buttons that lead to the different game scenes  
[TEMPORARY SCENES TO TEST/DEBUG]  
(Gameplay Test, Character Controller Test, Block Generation Test, MIDI Parsing Test, Leaderboard Test, Results Scene Test, other testing scenes...).

Level generation

- Preset test level using non-randomised block patterns
- Generic block patterns used for balance  
(Plus shaped obstacle block with health in the middle [RISK/REWARD])
- Using MIDI to influence which block patterns are chosen and what blocks are used in the randomised block patterns

- (More snare hits allow for higher amounts of single non-joined blocks, less would have more joined block colours to allow for less hits needed to be used to move downwards through the level.

### Timeline scrolling

- Horizontal trigger collider scrolls down through the screen vertically in time with each measure of the music track. When it reaches the end it then spawns back at the top at the start of the measure.

### Input timings with MIDI track

- Test scene that prints the timings of the MIDI track for when inputs are determined to be in either the range of 'Too soon', 'Perfect', 'Too late' -> All considered 'On-beat'.
- 'Grace period' implementation to allow for it to be easier to hit the beats by allowing a small range of hitting the beat to be accepted. This also helps if the game slows down at any time causing inputs to be dropped.
- Allow to dynamically adjust latency based on the current framerate, this could also assist in dropped inputs if the game encounters slowdowns.
- Simple HUD element component that will be used in the finished gameplay scene: Beat counting display -> Set of sf::rectangle shapes that are aligned in a row. A rectangle shape's colour is changed from Blue to Red as the beat is aligned with said rectangle.

Example: Common time (4/4)

[Hit the first beat]

1, 2, 3, 4

**RED, Blue, Blue, Blue**

[Hit the second beat]

1, 2, 3, 4

**Blue, RED, Blue, Blue**

[Hit the third beat]

1, 2, 3, 4

**Blue, Blue, RED, Blue**

[Hit the final beat]

1, 2, 3, 4

**Blue, Blue, Blue, RED**

This will also make debugging visually clearer as to when the beat is synced with the MIDI's music. Using the MIDI parser, we can also change the amount of rectangles that represent the current time signature ( 3/4, 6/8, etc). This makes it easier for the user to understand when they should be hitting the key press.

### Collectibles:

- Basic spawning, collision and visuals (debug hitbox, basic sound cue on pickup).
- Item List:
  - Health (Record)
  - Bomb (Walk into to activate) (Destroys other blocks surrounding it)

- 1 UP (Adds to Lives Counter)
- Slow time (Beat Sweeper)
- Air Bubble (Stops health drain for a period of time)

First playtest

- Determines if the basic controls are intuitive and easy to remember when playing.
- Recording gameplay through screen capture to watch it back if any bugs appear and how to replicate them.
- General feedback for if the game is fun and interesting to play.

## **Phase 2: Simple prototype showcasing core gameplay (Weeks 3 - 4)**

First playable demo

November Presentation

Character controller polish

- Better collision detection for edge cases such as when the player is inbetween the space when two blocks from above are falling, which one damages the player.
- Also if the player is at the very edge of a block falling, allow the player to automatically jump to the side of the block IF that space is open (no other blocks there).

Options Menu

- Volume adjustment [Music, SFX]
- Difficulty [Adjust “Grace Period” range to make it more lenient or more strict

Block class:

- Sf::vector2f coordinates (location in the global gameplay space)
- Collider: ( Player + other blocks )
  - Sf::rectangle shape + sf::colour (debugging colours)
- Hitbox: ( Player [DAMAGE] )
  - Sf::rectangle shape + sf::colour (debugging colour)
- Block types:
  - 4 different colours: BLUE, PINK, GREEN, RED
  - Heavy - takes multiple hits to break + drain portion of health
- Texture/ Sprite:
  - 4 different colours: BLUE, PINK, GREEN, RED

Level Generation:

- SRand can be inefficient and not always generate a new number everytime it is ran.
- (every iteration we need to generate a new block section)
- We can use Mersenne Twister to not have this issue.
- We can also create an additional function to randomise this random number after it is generated, to ensure it a unique number everytime (i.e. generate another number and add or minus the original random number).

Second playtest:

- More bug finding.

#### MIDI Parse

- Convert from the older version of Big Endian to work with our current C++ code.
- Parse out string data so that we can get our BPM, Channel Name, Time Signature
- Calculate the distance between the notes. Use BPM and the time data where the note is stored in MIDI [MIN, SEC, MSEC].

#### Presentation:

- Showcase the small portion of work completed so far.
- Slides showing snippets of code to explain how it works in the current version of the project, and how it can change in the next phase of the project.
- How far is the core features completed (percentage along with pie charts).
- Using illustrations created for the existing documentation in the slides to explain certain ideas or how certain sections of code will work.

## **Phase 3: (Weeks 4 - 7) (Christmas Break)**

#### Tuning :

More polish for character tuning (Movement speed, falling speed) and bug fixing.

#### Visual Polish:

- Add visual polish such as particles for picking up Collectibles, disappear animation. Hit spark for when breaking a Block (different sprites/audio cues for if it is [invalid, too late, perfect, too soon]).
- Icons and different audio cues for when you chain a combo and how many blocks were in said combo chain.
- Background art (sprite) for different scenes (Main Menu, Gameplay, Options, etc)

#### Joystick Controller Inputs:

- Allow for inputs from a controller (DS4, Xinput, etc).

#### Third playtest:

- More bug finding.

(After Christmas)

## **Phase 4:**

## **Phase 5: [FINAL PHASE]**