**Test Plan:**

**Level Load:**

* **Preconditions:**
  + None
* **Steps:**

1. Launch the game (./TIMELOCK)

* **Expected Result:**
  + The demo level is loaded. This should contain the following elements:
    - Background (2 background layers + foreground chains)
    - Boundaries (thin 1px border)
    - Camera created
    - Player
    - Any platforms specified in the level load function.

**Movement:**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a static platform below the spawn point of the player.
* **Steps:**

1. Press the **left arrow key**
2. Press the **right arrow key**
3. Press the **up arrow key**
4. Hold the **left arrow key**, then hold the **right arrow key**
5. Hold the **left arrow key**, then press **up arrow key** while holding **left arrow key**
6. Hold the **left arrow key**, then press **up arrow key** while releasing **left arrow key**
7. Hold the **up arrow key,** then press the **left arrow key**

* **Expected Result:**
  + The character will move. The following movements should happen:

1. The character moves left until the **left arrow key** is released. They will accelerate at PLAYER\_WALK\_ACCELERATION px/s until they reach PLAYER\_MAX\_WALKING\_SPEED.
2. The character moves right until the **right arrow key** is released. They will accelerate at PLAYER\_WALK\_ACCELERATION px/s until they reach PLAYER\_MAX\_WALKING\_SPEED.
3. The player jumps exactly once. Holding the **up arrow key** or pressing it in the air will not cause the player to jump higher. They will move upwards at JUMP\_VELOCITYpx/s, and be slowed down by GRAVITY px/s.
4. The player will begin moving left, and once the right arrow key is pressed, they will quickly change directions and begin moving right.
5. The player will move according to (1) until the **up arrow key** is pressed. Once the **up arrow key** is pressed, the player will move according to (3) while maintaining leftward velocity. (horizontal distance covered while in-air is dependent on how long left arrow key held before jumping (maximized when max walking speed achieved before jumping)
6. The player will move according to (1) until the **up arrow key** is pressed. Once the **up arrow key** is pressed, the player will move according to (3) while maintaining leftward velocity. Once the **left arrow key** is released, the player will halt their horizontal movement and continue with the vertical motion until they reach the ground.
7. The player will move according to (3) until the **left arrow key** is pressed. Once the **left arrow key** is pressed, the player will move according to (1) mid-air until the player reaches the ground. Displacement to the left of their original jump position is directly related to how quickly the left arrow key is pressed (highest displacement when pressed soon after jump)

**Game State:**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
* **Steps:**

1. Press the **R** key
2. Walk in to the boundary (thin 1px border at the edge of the level)

* **Expected result:**
  + After both steps, the level should restart. This means
    - All states will return to the original state of the level. Any moving platforms will reset, and any physics objects will return to their initial position.
    - If time control is active, it will reset to normal. All time controllable objects will return to their original velocity.

**\*\*\* NOTE: the time control tests do not cover cooldowns at the moment as we have set this to be very low for M1 playtesting. Cooldowns will be more relevant when we get into more detailed level design.**

**Time Control (Decelerate - Activate):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There must be a time controllable object in the level (ie. a moving platform)
* **Steps:**

1. Press the **W** key
2. Wait DECELERATION\_DURATION\_MS (currently 10s)

* **Expected Results:**
  + On step 1, the time decelerate ability will activate. The following should happen:
    - The blue vignette screen shader will fade in over DECELERATION\_EMERGE\_MS (currently 150ms)
    - Objects with the time controllable component will transition smoothly to slow their velocity down over DECELERATION\_EMERGE\_MS (currently 150ms)
      * Precisely, they should move at DECELERATE\_FACTOR x the original speed. (currently 0.2x)
    - Objects will remain slowed down for the duration DECELERATION\_DURATION\_MS (currently 10s)
      * After when the ability deactivates, the objects will smoothly transition to normal speed and the screen shader will fade out again over DECELERATION\_EMERGE\_MS

**Time Control (Accelerate - Activate):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There must be a time controllable object in the level (ie. a moving platform)
* **Steps:**

1. Press the **Q** key
2. Wait ACCELERATION\_DURATION\_MS (currently 10s)

* **Expected Results:**
  + On step 1, the time accelerate ability will activate. The following should happen:
    - The red vignette screen shader will fade in over ACCELERATION\_EMERGE\_MS (currently 150ms)
    - Objects with the time controllable component will transition smoothly to speed their velocity down over ACCELERATION\_EMERGE\_MS (currently 150ms)
      * Precisely, they should move at ACCELERATE\_FACTOR x the original speed. (currently 2x)
    - Objects will remain slowed down for the duration ACCELERATE\_DURATION\_MS (currently 10s)
      * After when the ability deactivates, the objects will smoothly transition to normal speed again and the screen shader will fade out over ACCELERATION\_EMERGE\_MS

**Time Control (Decelerate - Deactivate):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There must be a time controllable object in the level (ie. a moving platform)
* **Steps:**

1. Press the **W** key
2. Before DECELERATION\_DURATION\_MS has elapsed, press the **W** key again

* **Expected Results:**
  + On step 1, the time decelerate ability will activate. Please see the decelerate activate test for the expected behaviour.
  + On step 2, the time decelerate ability will deactivate. The following should happen:
    - The objects will smoothly transition to normal speed again and the screen shader will fade out over DECELERATION\_EMERGE\_MS

**Time Control (Accelerate - Deactivate):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There must be a time controllable object in the level (ie. a moving platform)
* **Steps:**

1. Press the **Q** key
2. Before ACCELERATE\_DURATION\_MS has elapsed, press the **Q** key again

* **Expected Results:**
  + On step 1, the time accelerate ability will activate. Please see the accelerate activate test for expected behaviour.
  + On step 2, the time decelerate ability will deactivate. The following should happen:
    - The objects will smoothly transition to normal speed again and the screen shader will fade out over ACCELERATION\_EMERGE\_MS

**Time Control (Decelerate to Accelerate):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There must be a time controllable object in the level (ie. a moving platform)
* **Steps:**

1. Press the **W** key
2. Before DECELERATION\_DURATION\_MS has elapsed, press the **Q** key

* **Expected Results:**
  + On step 1, the time decelerate ability will activate. Please see the decelerate activate test for expected behaviour.
  + On step 2, the time accelerate ability will activate. Please see the accelerate activate test for expected behaviour.

**Time Control (Accelerate to Decelerate):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There must be a time controllable object in the level (ie. a moving platform)
* **Steps:**

1. Press the **Q** key
2. Before ACCELERATION\_DURATION\_MS has elapsed, press the **W** key

* **Expected Results:**
  + On step 1, the time accelerate ability will activate. Please see the accelerate activate test for expected behaviour.
  + On step 2, the time decelerate ability will activate. Please see the decelerate activate test for expected behaviour.

**The following two tests should be run with time accelerate and decelerate in different combinations. The expected results are the same, but context dependent on the speed of the platform when time is sped up / slowed down:**

**Physics (Player -> Static Platform)**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a static platform below the spawn point of the player.
  + There is a static and tall platform to the right of the player, touching the horizontal platform
* **Steps:**

1. Wait until the player lands on the platform
2. Jump a few times
3. Move to the edge of the platform (left)
4. Run to the right into the vertical platform
5. Hold the right run button to continue running into the vertical platform
6. Run and jump to the right into the vertical platform

* **Expected Behaviour:**

1. The player will land on the platform and not pass through
2. After jumping, the player will land on the platform and not pass through
3. The player will remain perpendicular to the floor of the platform until both feet of the player are off the platform (ie no contact between the two)
4. The player will collide with the wall and stop running.
5. Even when holding the run button, they will not pass through
6. Even when running and jumping into the platform, they will not pass through

**Physics (Player -> Horizontal Moving Platform)**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a static platform below the spawn point of the player.
  + There is a fast moving horizontal platform, and a slow moving one.
* **Steps:**

1. Run and jump onto the slow moving horizontal platform
2. Jump off of the slow moving platform.
3. Run and jump onto the fast moving horizontal platform
4. Jump off of the fast moving platform.

* **Expected Results:**

1. When landing on either platform, the player will accelerate to match the velocity of the platform slowly. The player should slide a little bit on top of the platform before reaching the desired speed.
2. When the player jumps off of the platforms, they should maintain their velocity and slowly come to a stop
   1. (in the air, slowing down by AIR\_RESISTANCE px/s, or if landing on another platform slowing down by STATIC\_FRICTION px/s, flings are hard to control so they are not explicit tests, but make sure to try until you get both cases).
   2. The fast platform should fling the player.

**Physics (Player -> Vertical Moving Platform)**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a static platform below the spawn point of the player.
  + There is a fast moving vertical platform, and a slow moving one.
* **Steps:**

1. Run and jump onto the slow vertical moving platform
2. Jump off of the slow moving platform.
3. Run and jump onto the fast vertical moving platform
4. Jump off of the fast moving platform.

* **Expected Results:**

1. When landing on either platform, the player will accelerate to match the velocity of the platform slowly.
   1. If the vertical platform is moving up, it will push the player up with it
   2. If the vertical platform is moving down, the player will fall at the speed of gravity (if the platform is moving faster than PLAYER\_MAX\_FALLING\_SPEED, the player will lose contact with the platform.
2. When the player jumps off of the platforms, they should maintain their velocity and slowly come to a stop
3. (in the air, slowing down by AIR\_RESISTANCE px/s, or if landing on another platform slowing down by STATIC\_FRICTION px/s, flings are hard to control so they are not explicit tests, but make sure to try until you get both cases).
4. The fast platform should fling the player.

**Physics (Player -> Object):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a static platform below the spawn point of the player.
  + There is a physics object spawning directly above the player
* **Steps:**

1. Do not move and allow the block to land on your head.
2. Jump with the block on your head.
3. Walk so that the physics object falls off your head.
4. Walk into the physics object, and continue walking
5. Jump on top of the physics object.
6. Push the physics object off the platform.

* **Expected Results:**

1. The block will land on top of the player, it will not pass through or move to the ground, remaining on their head
2. The block will move up with the player and fall with the same gravity acceleration.
3. The block will move slightly with the player, but slide off pretty easily
4. The block will be pushed by the player. The player will not phase through the block.
5. The player will land on top of the block
6. The object will fall off the platform with gravity.

**AI (Canon Tower: Approach, Fired, Normal Time):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a canon tower created in the world, approachable by the player
  + There is enough space for the player to move (at least CANON\_TOWER\_DETECTION\_RANGE from canon tower)
* **Steps:**

1. Start at position away from the tower at a distance over CANON\_TOWER\_DETECTION\_RANGE.
2. Approach the tower such that the distance between the player and the tower falls below CANON\_TOWER\_DETECTION\_RANGE.
3. Move around the tower, ensure still within the range.

* **Expectations:**
  1. When at distance below CANON\_TOWER\_DETECTION\_RANGE, the canon starts to rotate its barrel towards the player, along the direction leading to smaller rotation angle.
  2. As player moves around, the barrel keeps tracing the player as long as the player is inside the range.
  3. After CANON\_TOWER\_AIM\_TIME\_MS ms, the tower barrel shrinks, thrusts forward along its aimed direction and fires a bolt projectile with red silhouette; during this event, the barrel does not change angle.
  4. Once fired, the bolt pursues a parabolic path, and disappears when colliding with the player, platforms or other bolts.
  5. Once the player collides with the bolt, the player is killed and respawns.
  6. Once fired, the barrel shrinks back to normal length. During this event, the barrel does not change its angle.
  7. After returning to normal length, the canon tower starts to aim again.

**AI (Canon Tower: Approach, Fired, Accelerated Time):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a canon tower created in the world, approachable by the player
  + There is enough space for the player to move (at least CANON\_TOWER\_DETECTION\_RANGE from canon tower)
* **Steps:**

1. Activate time acceleration.
2. Start at position away from the tower at a distance over CANON\_TOWER\_DETECTION\_RANGE.
3. Approach the tower such that the distance between the player and the tower falls below CANON\_TOWER\_DETECTION\_RANGE.
4. Move around the tower, ensure still within the range.

* **Expectations:**
  1. Same expectations as case above, but the canon rotation speed and barrel animation speed are visibly faster.

**AI (Canon Tower: Approach, Fired, Decelerated Time):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a canon tower created in the world, approachable by the player
  + There is enough space for the player to move (at least CANON\_TOWER\_DETECTION\_RANGE from canon tower)
* **Steps:**

1. Activate time deceleration.
2. Start at position away from the tower at a distance over CANON\_TOWER\_DETECTION\_RANGE.
3. Approach the tower such that the distance between the player and the tower falls below CANON\_TOWER\_DETECTION\_RANGE.
4. Move around the tower, ensure still within the range.
5. Ensure time deceleration continues until bolt projectile fired.
6. Interact with the projectile until deceleration effect ends.
7. Repeat step 2-4; ensure when the projectile is fired, time control state is in normal state.
8. While the projectile is fired, activate time deceleration.

* **Expectations:**
  1. Before the bolt is fired, same expectations as case above, but the canon rotation speed and barrel animation speed are visibly slower.
  2. When the bolt is fired during time deceleration, it has green silhouette, does not disappear, and the player can safely interact with the bolt. The bolt behaves as a normal physics object.
  3. When the bolt is fired during normal time speed, it has red silhouette. Once time deceleration is activated, it changes from red to green silhouette.

**AI (Canon Tower: Approach, Depart):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a canon tower created in the world, approachable by the player
  + There is enough space for the player to move (at least CANON\_TOWER\_DETECTION\_RANGE from canon tower)
* **Steps:**

1. Start at position away from the tower at a distance over CANON\_TOWER\_DETECTION\_RANGE.
2. Approach the tower such that the distance between the player and the tower falls below CANON\_TOWER\_DETECTION\_RANGE.
3. Before the barrel shrinks (before entering the range for CANON\_TOWER\_AIM\_TIME\_MS ms), leave the range.
4. Repeat step 1 and 2; after the barrel starts to shrink, leave the range.

* **Expectations:**
  1. When the player leaves the range before the barrel shrinks, the barrel remains unchanged. No further actions are taken by the tower.
  2. When the player leaves after the barrel shrinks, a projectile is fired along the barrel’s angle before it shrinks.

**Camera:**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a static platform below the spawn point of the player.
* **Steps:**

1. Move left/right
2. Jump

* **Expected results:**
  + The camera should follow the player, catching up to its position.
  + If the player jumps fast enough (launched by a platform), they may slightly leave the camera’s FOV. It will catch up when the player stops/slows down.

**Render (Parallax):**

* **Preconditions:**
  + The game has been launched, and the level has been loaded. (Create player and create camera must have been called)
  + There is a static platform below the spawn point of the player.
* **Steps:**

1. Move in any direction (up, down, left , right)

* **Expected results:**
  + Each background layer should move **opposite** to the player’s direction of movement according to the following relative speeds (layers ordered in increasing distance to camera):
    - Foreground (chain) should move faster than SOP (speed of player)
    - Midground should move at the same speed SOP
    - Gold/metal background should move slightly slower than SOP
    - Brown/gear background should move the slowest in comparison to SOP