Test Plan

Updated 2025-03-13

Terms

i-frames: “Invincibility Frames” refers to the duration of invincibility after the player takes damage.

Testing Strategy

The primary testing method has been integration testing of system interaction (e.g. player taking damage, ghost possession, etc) by creating “debug rooms” that include the relevant systems for observation.

Periodic user testing has been done with a static “test level” that users are asked to complete. The users are observed over the duration of their playtime. Any questions they have are answered and recorded. The playtester is then asked a series of questions about their experience and asked to provide any suggestions.

Unit testing will be implemented using the Godot Unit Testing (GUT) library for core systems such as the player/ghost state machines and possessable objects. This will ensure they are entering and exiting the various states with the appropriate internal values set or unset at appropriate times.

1. Unit Testing

1.1 Player States

1.1.1 Player Enter LIVING State

Precondition: The player is not in LIVING state
Postcondition:

- The player is in LIVING state

- LIVING state variables correctly set
- State change signal emitted

Procedure:

- Set player to LIVING state

Results:

1.1.2 Player Exit LIVING State

Precondition: The player is in LIVING state

Postcondition: The Player is not in LIVING state

Procedure:

- Set player to DYING state

Results:

1.1.3 Player Enter DYING State

Precondition: The player is not in DYING state

Postcondition:

- The player is in DYING state
- DYING state variables correctly set
- State change signal emitted

Procedure:

- Set player to DYING state

Results:

1.1.4 Player Exit DYING State

Precondition: The player is in DYING state

Postcondition:

- The player is not in DYING state
- DYING state variables set to default values

Procedure:

- Set player to DEAD state

Results:

1.1.5 Player Enter DEAD State

Precondition: The player is not in DEAD state

Postcondition:

- The player is in DEAD state

- DEAD state variables correctly set
- State change signal emitted

Procedure:

- Set player to DEAD state

Results:

1.1.6 Player Exit DEAD State

Precondition: The player is in DEAD state

Postcondition:

- The player is not in DEAD state
- DEAD state variables set to default value

Procedure:

- Set player to LIVING state

Results:

1.2 Ghost States

1.2.1 Ghost Enter WAITING State

Precondition: The ghost is not in WAITING state

Postcondition:

- The ghost is in WAITING state
- WAITING state variables correctly set

Procedure:

- Set ghost to WAITING state

Results:

1.2.2 Ghost Exit WAITING State

Precondition: The ghost is in WAITING state

Postcondition:

- The ghost is not in WAITING state
- WAITING state variables set to default values

Procedure:

- set ghost to POSSESSING state

Results:

1.2.3 Ghost Enter POSSESSING State

Precondition: The ghost is not in POSSESSING state
Postcondition:

- The ghost is in POSSESSING state
- POSSESSING state variables correctly set

Procedure:

- Set ghost to POSSESSING state

Results:

1.2.4 Ghost Exit POSSESSING State

Precondition: The ghost is in POSSESSING state
Postcondition:

- The ghost is not in POSSESSING state
- POSSESSING state variables set to default values.

Procedure:

- Set ghost to WAITING state

Results:

1.2.5 Ghost Enter STUNNED State

Precondition: The ghost is not in STUNNED state
Postcondition:

- The ghost is in STUNNED state
- STUNNED state variables correctly set

Procedure:

- Set ghost to STUNNED state

Results:

1.2.6 Ghost Exit STUNNED State

Precondition: the ghost is in STUNNED state
Postcondition:

- the ghost is not in STUNNED state
- STUNNED state variables set to default values

Procedure:

- set ghost to WAITING state

Results:

1.2.7 Ghost Enter ATTACKING State

Precondition: The ghost is not in ATTACKING state
Postcondition:

- The ghost is in ATTACKING state
- ATTACKING state variables correctly set

Procedure:

- Set ghost to ATTACKING state

Results:

1.2.8 Ghost Exit ATTACKING State

Precondition: the ghost is in ATTACKING state
Postcondition:

- the ghost is not in ATTACKING state
- ATTACKING state variables set to default values

Procedure:

- set ghost to WAITING state

Results:

1.2.9 Ghost Enter MOVING State

Precondition: The ghost is not in MOVING state
Postcondition:

- The ghost is in MOVING state
- MOVING state variables correctly set

Procedure:

- Set ghost to MOVING state

Results:

1.2.10 Ghost Exit MOVING State

Precondition: the ghost is in MOVING state
Postcondition:

- the ghost is not in MOVING state
- MOVING state variables set to default values

Procedure:

- set ghost to WAITING state

Results:

1.3 Possessables

1.3.1 Possessable Possessed

Precondition: Object is not possessed

Postcondition:

- Object is possessed
- possession-related variables set

Procedure:

- possess the object

Results:

1.3.2 Possessable Depossessed

Precondition: Object is possessed

Postcondition:

- Object is depossessed
- possession-related variables are set to post-possession state
- After post-possession time expires, possession-related variables are set to default values

Procedure:

- Depossess the object
- Wait for post-possession timer to expire

Results:

1.3.3 Attack Possessable Possessed

Precondition: Object is not possessed

Postcondition:

- Object is possessed
- possession-related variables set

Procedure:

- possess the object

Results:

1.3.4 Attack Possessable Depossessed

Precondition: Object is possessed

Postcondition:

- Object is depossessed
- possession-related variables are set to post-possession state
- After post-possession time expires, possession-related variables are set to default values

Procedure:

- Depossess the object
- Wait for post-possession timer to expire

Results:

1.3.5 Ranged Attack Possessable Possessed

Precondition: Object is not possessed

Postcondition:

- Object is possessed
- possession-related variables set

Procedure:

- possess the object

Results:

1.3.6 Ranged Attack Possessable Depossessed

Precondition: Object is possessed

Postcondition:

- Object is depossessed
- possession-related variables are set to post-possession state
- After post-possession time expires, possession-related variables are set to default values

Procedure:

- Depossess the object
- Wait for post-possession timer to expire

Results:

1.3.7 Throwable Attack Possessable Possessed

Precondition: Object is not possessed

Postcondition:

- Object is possessed
- possession-related variables set

Procedure:

- possess the object

Results:

1.3.8 Throwable Attack Possessable Depossessed

Precondition: Object is possessed

Postcondition:

- Object is depossessed
- possession-related variables are set to post-possession state
- After post-possession time expires, possession-related variables are set to default values

Procedure:

- Depossess the object
- Wait for post-possession timer to expire

Results:

2. Integration Testing

2.1 Player Hurtbox Collision Detection

2.1.1 Player Enter Static Hurtbox

Precondition: player i-frames are not active
Postcondition: player i-frames are active and the player has taken one unit of damage
Procedure:

- Create a room with one hurtbox
- Have the player enter the hurtbox; the player should receive one unit of damage and activate i-frames, receiving no further damage during the i-frame window

Results: PASSED - the player received one unit of damage upon entering the hurtbox and did not receive any further damage during the i-frame window

2.1.2 Player Enter and Re-enter Static Hurtbox

Precondition: player i-frames are not active
Postcondition: player i-frames are active and the player has taken one unit of damage
Procedure:

- Create a room with 1 hurtbox
- Have the player enter the hurtbox; the player should receive one unit of damage and activate i-frames
- While i-frames are active, have the player exit and re-enter the hurtbox; the player should receive no damage

Results: PASSED - the player did not receive another unit of damage after re-entering the hurtbox during the i-frame window

2.1.2 Player Remain in Static Hurtbox

Precondition: player i-frames are not active
Postcondition: player i-frames are active and the player has take two units of damage
Procedure:

- Create a room with 1 hurtbox
- Have the player enter the hurtbox and remain inside the hurtbox collision area
- The player should receive one unit of damage and activate i-frames; the player should take no further damage
- After the i-frames elapse, the player should receive another unit of damage and activate i-frames

Results: PASSED - the player received a second unit of damage after the i-frame window elapsed and reactivated the i-frames window

2.1.3 Player Enter Second Static Hurtbox While Already Within Hurtbox

Precondition: player i-frames are not active
Postcondition: player i-frames are active and the player has take one unit of damage
Procedure:

- Create a room with 2 or more overlapping hurtboxes
- Have the player enter a hurtbox; the player should take one unit of damage and activate i-frames
- While i-frames are active and the player is inside the first hurtbox area, have the player enter the second overlapping hurtbox
- The player should not receive another unit of damage

Results: PASSED - the player did not receive another unit of damage when entering the second hurtbox during the i-frame window

2.1.4 Player Remain in Overlap Between Static Hurtboxes

Precondition: player i-frames are not active
Postcondition: player i-frames are active and the player has taken two units of damage
Procedure:

- Create a room with 2 or more overlapping hurtboxes
- Have the player enter one hurtbox
- The player should receive one unit of damage and activate i-frames
- During the i-frames, have the player enter the area where both hurtboxes overlap; the player should take no damage
- After the i-frames elapse, the player should receive only one unit of damage while contacting two hurtboxes and activate i-frames

Results: PASSED - the player only received one unit of damage while remaining in contact with multiple hurtboxes

2.2 Ghost-Possessable Interaction

2.2.1 Single Ghost, Single Possessable, Player Far Away

Precondition:

- The player is not in DEAD state
- The player is not within the attack range of the possessable

Postcondition:

- The ghost depossesses the object
- The possessable object is available to be re-possessed

Procedure:

- Create a room with a single ghost and a single possessable
- Wait for the ghost to possess the object
- Wait for the ghost to depossess the object
- Wait for the ghost to repossess the object

Results: PASSED - the ghost was able to possess and repossess the object

2.2.2 Single Ghost, Single Possessable, Player Nearby

Precondition:

- The player is not in DEAD state
- The player is within attack range of the possessable

Postcondition:

- The ghost depossesses the object
- Alternatively, the ghost attacks the player - results in depossession
- The possessable object is available to be repossessed

Procedure:

- Create a room with a ghost and possessable
- Have the player enter the attack range of the possessable
- Wait until the ghost possesses the object
- Wait until the ghost decides an action; ghost should depossess the object either on its own or after attacking the player

Results: PASSED - the ghost was able to possess and repossess the object indefinitely, sometimes attacking the player

2.2.3 Player Pushes Targeted Possessable

Precondition:

- The player is not in DEAD state
- A ghost is targeting a possessable but has not reached it yet

Postcondition:

- The ghost tracks the changing possessable position and eventually reaches it

Procedure:

- Create a room with a ghost and a possessable
- Wait for the ghost to target the possessable
- Have the player push the possessable to a different location before it is possessed
- The ghost should track the position of the possessable as it moves and be able to possess it on contact

Results: PASSED - the ghost tracks the position of the targeted possessable as it moves

2.2.4 Player Pushes Possessed Possessable

Precondition: The player is not in DEAD state

Postcondition:

- The object remains possessed
- The ghost follows the possessable object
- The possessable is still capable of attacking the player if in range

Procedure:

- Create a room with a single ghost and a single possessable
- Wait for the ghost to possess
- Have the player push the object once possessed
- The ghost possessing the object should still be able to accurately attack the player or depossess the object

Results: PASSED - the ghost remained in possession of the object and followed it as it was moved while possessed

2.2.5 Many Ghosts, Single Possessable

Precondition: The player is not in DEAD state

Postcondition:

- The object remains possessable by all ghosts after depossession
- All simultaneously targeting ghosts enter WAITING state upon the first ghost possessing the object

Procedure:

- Create a room with many ghosts and a single possessable
- Wait until more than one ghost targets the possessable while it is not possessed

- The first ghost who reaches the shared target should possess it and all other ghosts with the same target should enter the WAITING state since there are no other possessables available

Results: PASSED - whenever multiple ghosts targeted the same possessable, the first ghost to reach it would possess it and all other targeting ghosts returned to WAITING state

2.2.6 Two Ghosts, Two Possessables

Precondition: The player is not in DEAD state

Postcondition:

- The object remains possessable by all ghosts after depossession
- The non-possessing ghost re-targets the other possessable

Procedure:

- Create a room with two ghosts and two possessables
- Wait for both ghosts to target the same possessable
- The first ghost to reach the shared target should possess it and the other ghost with the same target should retarget the other available possessable

Results: [TO BE COMPLETED]

2.2.7 Many Ghosts, Many Possessables

Precondition: The player is not in DEAD state

Postcondition:

- All objects remain possessable
- All ghosts maintain regular WAITING->POSSESSING->WAITING etc behavior

Procedure:

- Create a room with many ghosts and many possessables
- Wait for a period of time to ensure all ghosts maintain standard behavior (i.e. no freezing, no multi-possessions, etc)

Results: PASSED - all ghosts maintained the standard behavior of transitioning between WAITING and POSSESSING for the observation duration

2.3 Shrines and Revival

2.3.1 Default Shrine Activated at Start of Level

Precondition: N/A

Postcondition: Default shrine is activated

Procedure:

- Start a level

- Move to default shrine room and observe the shrine

Results:

2.3.2 Default Shrine Does Not Consume On Revival

Precondition: Player not in DEAD state

Postcondition: Default shrine is not consumed

Procedure:

- Start a level
- Change player state to DEAD state with default shrine as the closest active shrine

Results:

2.3.3 Shrine Activation

Precondition:

- Player not in DEAD state
- Non-default shrine is inactive

Postcondition:

- Shrine is active

Procedure:

- Start a level with a non-default shrine
- Activate the shrine

Results:

2.3.4 Shrine Revival and Consumption

Precondition:

- Player not in DEAD state
- Non-default shrine is active

Postcondition:

- Player is in DEAD state and at shrine location
- Shrine is consumed

Procedure:

- Start a level with a non-default shrine
- Activate the shrine
- Move player so the activated shrine is the nearest shrine
- Change the player state to DEAD

Results:

2.3.5 Shrine Activation While Consumed

Precondition:

- Player not in DEAD state
- Non-default shrine has been consumed

Postcondition:

- Non-default shrine remains consumed and inactive

Procedure:

- Start a level
- Activate the shrine
- Change the player state to DEAD, moving them to the shrine and consuming it
- Change the player state to LIVING or DYING
- Attempt to activate the shrine

Results:

2.3.6 Shrine Revival With Consumed Shrine

Precondition:

- Level contains at least two shrines
- One shrine is activated and the other has been consumed

Postcondition:

- The player is revived at the non-consumed shrine

Procedure:

- Start a level with at least two shrines
- Activate both shrines
- Revive at and consume one shrine
- Move the player in LIVING or DYING state to the consumed shrine
- Change the player state to DEAD

Results:

2.4 Ghost Attack

2.4.1 Ghost Attack in Dying State

Precondition: The player is in

Postcondition:

- All objects remain possessable
- All ghosts maintain regular WAITING->POSSESSING->WAITING etc behavior

Procedure:

- Create a room with many ghosts and many possessables

- Wait for a period of time to ensure all ghosts maintain standard behavior (i.e. no freezing, no multi-possessions, etc)

Results: PASSED - all ghosts maintained the standard behavior of transitioning between WAITING and POSSESSING for the observation duration

3. User Testing

There is a **playtesting report** to be filled out by the player and/or test observer after the playtest, including game version and playtester information, sentiment, and questions or suggestions. Responses to the form are tracked in a spreadsheet. When starting the game, playtesters will be presented with a short “how to play” instruction set that describes the mechanics of the game before starting the game.

Link to the user testing spreadsheet