Unit Tests:

Test	User Input	Expectation	Outcome
Player Movement (keyboard)	WASD, Arrow Keys	Player is moved as per unit vector in direction multiplied by speed	This happens
Player Movement (USB gamepad)	Gamepad d-pad (discrete axis values)	Player is moved as per unit vector in direction multiplied by speed	This happens
Player Collision with Interactive	N/A (irrelevant)	The object that inherits from Interactive triggers its own collide() method when its rect intersects with the Player's rect	This happens
Player Collision with Blocker	WASD, Arrow Keys, Gamepad d-pad	Player is moved back (vector opposing previous movement) and remains in same position overall	The vertical and horizontal components sometimes do not respectively cancel out, leading to net movement of the Player along the side a Blocker-inheriting object, opposing the direction of input motion
Reload rooms	Middle mouse button, Button 9 on gamepad	The game is frozen, and the save state selector messagebox dialogue is triggered, after which the selected save state is loaded, and the game continues as normal (that is, the gameplay remains constant even after successive state loading)	This happens
Music selecting and cycling	N/A	Once the defined end event for the background music channel is triggered, a file in the Sounds/Music/ directory is randomly chosen and played on the music channel, and its name is logged to console.	This happens, though the game freezes slightly when starting a new track
Music pausing and skipping	Gamepad buttons 4, 5	The music that is playing is paused/unpaused or stopped, respectively.	This happens, but pausing sometimes causes the speakers to crackle
Player response to flip	N/A	This isn't a subroutine, but rather an	It works as intended

		alteration to the player's flip attribute that reverses Vector component input and flips the icon() for placement.	
Room Switching	N/A	This isn't a subroutine, but rather an alteration to the Game object's room_pos attribute that leads to different elements being called from the levels dictionary.	It works as intended, with the player being able to move between rooms as though they are one large connected space.
Trigger triggering targets	N/A	A Trigger object triggers its target Wall's toggle() function when the Trigger collides with a Player as per its parent Interactive. This leads to target Walls being toggled on or off as required, and therefore either blocking or not interacting with the player, respectively	This is exactly what happens, the Trigger triggers its target's toggle function, which either kills or repairs its rect_check to either ignore or stop ignoring the Player.
Boosts boosting the Player	WASD/Arrow Keys/Gamepad	When the player collides with the boost as per Interactive, the boost triggers a while loop for x frames, where x is (double over speed) the distance that the player is boosted (eg if boosting the player 4*speed, this involves 8 frames of the player moving continuous half speed)	This occurs, though any perpendicular motion is seemingly lost, and for unclear reasons, the exact boost seems to vary slightly in regards to when the boost triggers.
Save State Loader (debug)	mouse/keyboard	A highly invaluable debug system, the save state loader is intended to allow the player to select (from a set of defined save states) a save state to load, effectively continuing the game from that point as though they had completed the previous sections of the game as well. SDL2's messagebox capabilities are called via pygame to provide the user with an input mechanism.	This system works very well, despite the fact that pygame._sdl2 is experimental

Alongside this, continuous testing was employed during the construction of each room, as the placement of each new Element can not be observed prior to a reload of the game, and so the game was reloaded after every new entry to the levels config file to ensure correct placement.

User Experience testing

In my extensive testing of the game, I noticed that the objectives of the game were not clear, and that gameplay was often clunky, adding unnecessary difficulty to the otherwise easy game. The lack of challenging puzzle is largely due to time constraints, as my sketched level design plan included a fourth row, with two hidden buttons corresponding to two walls blocking the path to the exit, adding a further level of puzzle to the otherwise simple and highly linear gameplay. I had also planned to add text, cutscenes, and/or screens that would have conveyed essential information to the player as to the premise and objective(s) of the game, but I was ultimately unable to include these in a suitable manner. As it is now, players should still be able to figure out how to play the game and what to do, though in some areas, this may be unclear, such as in the interactions of Triggers and their target Walls. I was unable to get other systems or people to test my project with, but I have confirmed that the product works on my own computer.

Comparison to Original design specs

The final product is almost completely different from the proposed project, as I changed my mind several times during the development process, due to unexpected challenges and loss of interest in ideas. My original idea would have been too complex for me to individually produce a satisfactory product centered around that idea, so I decided to change my idea to something simpler, but still unique. This is how I came to the idea of a top-down game with the player having to unlock movement in each direction, which is a radically different idea from a card game based RPG. By being able to focus upon the geometry of movement, and not having to also worry about card combat, I was able to produce a more developed product, though I am still unsatisfied with its lack of completion.