TEST PLAN – ALL IN

| In your /doc folder, upload a test plan that outlines how to user-test your game, title the document: **test-plan.docx**. For example, what keys to press or what mouse actions are necessary. This short document will be used by the TAs to test and mark your game. |
| --- |

# Basic overview of the game

* WASD controls (W for up, A for left, S for down, D for right)
* Auto shooting
* The player (called protagonist) always faces the mouse regardless of movement direction

# Basic User controls and visuals

* Load the game
* Use WASD to move: (W for up, A for left, S for down, D for right)
* Move the mouse around the protagonist and see the protagonist face the mouse in 4 directions (up, down, left, right)
* Notice that the protagonist automatically shoots three types of projectiles at the mouse
  + Roulette ball (yellowish small ball)
  + Card
  + Dart (most infrequent)
* The protagonist’s projectiles do damage and eventually enemies.

# Basic environment

* Enemies, which are cards, spawn in the ‘room’. There is only one type of enemy called King Clubs.
* Enemies (King Clubs) slowly move toward the player
* Enemies (King Clubs) only spawn in the room.
* The protagonist is enclosed in a room (surrounded on all four sides by walls)
  + Travel in a direction till you meet a wall.
  + The protagonist is not able to pass through the wall
  + No enemies spawn outside the wall.

# Thorough test for each criterion

Textured geometry

* Roulette balls that are shot by the protagonist have textured geometry
* The walls use textured geometry.

Basic 2D transformations

* Notice projectiles the player shoots use translation transformation
* Notice enemy King Clubs uses translation transformation
* Notice dart projectile uses rotation to always point to the mouse when fired by the protagonist

Key-frame/state interpolation

* Notice poop weapon uses state interpolation

Keyboard/mouse control

* Keyboard control: Use WASD to move. The player changes position (on the map instead of on-screen)
  + Each key moves the player in the respective direction. Press two non-contradictory keys to move diagonally
* Mouse control: The player changes texture to face towards the mouse in 4 possible directions (up, down, left, right)
  + Move the mouse around the player to see player’s texture change

Random/coded action

* Hardcoded action: Enemy king clubs is hardcoded to slowly travel to the player’s latest position.
  + To check, see Enemy King clubs move towards the player, even when the player is moving around.

Well-defined game-space boundaries

* The player is surrounded by walls on four sides, and the player cannot walk forever out of the level or fall out of the level
  + To find the walls, travel using WASD in a specific direction until you find the wall
  + When the player interacts with the wall, there is no jittery, and the player cannot clip out of the wall
* Enemies do not spawn outside the wall.
* Projectiles cannot pass out of the wall (disappear upon wall contact). To verify, shoot at the wall.

Simple collision detection & resolution (e.g. between square sprites)

* Collision between projectiles and enemies makes enemies take damage and can end up killing enemies.
* The collision between the enemy and the player will kill the player (the game will fade out and restart)
* The collision between the player and the wall will ensure the player does not move out of the wall.

Stable frame rate and minimal game lag.

* Buttery smooth. You see it.

No crashes, glitches, or unpredictable behaviour.

* All crashes are planned, and we have none planned.

# Creative elements: Two BASIC features

* Our implementation has 2 basic features: Reloadability, and Camera controls

Camera controls

* We choose a camera that follows the player around. To see this in action: when the player uses WASD to move, the camera follows the player and the world moves around them.

Reloadability

* Pressing the escape key saves the game state and quits the game. Re-opening the game by running the executable reloads the game from the saved game state. The player position, enemy (king clubs) position, projectile position and velocity, lerp\_projectiles position and velocity, and total points accumulated are saved and will be reloaded when the game is re-opened.