SKNAT

TEAM A5

Test Report

ALEXANDRE PORTUGAL - (1231727)
BELLA DUNOVSKA - (1250580)
MARTIN MIHOV - (1229174)
SAMUEL HILL - (1240663)
TENDAISHE NYAMAPFENI - (1251414)

1. Test Plan

At the beginning of our project we decided that we are splitting the work in different modules, which would be developed concurrently.

While developing each module, we did unit testing and functional tests to validate that the module was working correctly. We created JUnit tests that can be found in the junitTests package of our project. For the functional tests we wrote test case specifications and followed the use cases from our specifications document.

After making sure that the modules were working, we connected the game engine with the networking using the parsers that we have written. To test that the game was working properly with the modules connected, we did functional and integration tests.

To evaluate the final user experience and the HCI of our game we did usability testing.

2. Functional tests

Test Case ID	**Inputs**	**Outputs**		
		Expected	**Real**	**Pass?**
HealthPoints	1 hit from enemy	1 point decrement	1 point decrement	Yes
KeySpawn	1 tank brings the key to the objective	the key appeared on 170,210 (randomly)	the key appears at a new random position	yes
KeyPickup	1 tank moves over the key	the key moved along with the tank	the key is moving along with the tank	yes
KeyMechanics	1 tank brings the key to the objective	the key was reset at a new position and removed from the tank	the key falls from the tank	yes
Score	1 tank brings the key to the objective and holds it on it	a tank went over the objective for the specified time and then then an incremented amount of scores was printed out.	the keyholder's team receives a point	yes
AIGetKey	the key is at a random position on the field and one tank moves over the key.	the key starts to move along with the tank	the AI tank always strive to reach the key when it is not picked, so it gets the key.	Yes
AIFollowKeyHolder	one tank picks the key, the key moves along with it	the AI tank starts to follow the key holding tank	one tank is moving along with the key, the AI tank follows it precisely.	Yes

AIFollowObjective	AI tank is currently holding the key	the AI tank brings the key to the objective	the AI tank has collected the key and it reaches the objective	Yes
Tank Respawn	one tank loses all his health points and is killed	the tank appears at its initial position after five seconds	a tank from the opposite team shoots the tank three times which causes it to be respawned at its initial position after 5 seconds delay	Yes
AIShootAtOpponents	The AI tank is near an opponent tank	The AI tank shoots at the direction of the opponent tank	The AI tank shoots a bullet and hits the enemy tank.	Yes
AIAvoidObstacles	The AI tank is placed in a map with obstacles	The AI tank tries to reach its goal while avoiding obstacles and finding the shortest path.	The AI tank moves along the map avoiding obstacles and reaches its goal.	Yes
AIRotateGun	An AI tank and an opponent tank are nearby	The AI gun should rotate pointing to the direction of shooting	The AI tank produces a shot and the gun rotates at the direction.	Yes
AISearchShortestPath	An AI tank and a goal to reach.	The AI tank should do a search through the map and follow the found path.	The AI tank moves to the goal, following the shortest path	Yes

HealthPoints: Test whether a health is decremented after a player has been hit by a projectile.
Test items: Software code - checkForHits(), hitTank().
Input specifications: one hit from an enemy tank.
Output specifications: a decrement of one.
Environmental needs: at least two tanks from opposite team.

KeySpawn: Test whether a key is being reset at a new random position on the screen after it has been used for the main objective.
Test items: Software code - resetKey().
Input specifications: one tank brings the key to the objective.
Output specifications: the key appears at a new random position.
Environmental needs: one tank holding the key reaches the objective.

KeyPickup: Test whether a key is collected after a tank moves over it.
Test items: Software code - checkForKey().
Input specifications: one tank moves over the key.
Output specifications: the key starts to move along with the tank.
Environmental needs: one tank passing over the key.

KeyMechanics: Test whether a key is not being displayed alongside with the tank after it has scored an objective point.
Test items: Software code - checkForObjective(), resetKey().

Input specifications: one tank brings the key to the objective.
Output specifications: the key falls from the tank.
Environmental needs: one tank holding the key reaches the objective.

<u>**Score**</u>: Test whether a key is being brought over the objective for a specified amount of time and whether a team has received a point accordingly.
Test items: Software code - checkForObjective(), incrementTeamScores().
Input specifications: one tank brings the key to the objective and holds it on it.
Output specifications: the keyholder's team receives a point.
Environmental needs: one tank holding the key reaches the objective and stays there for the specified time.

<u>**AIGetKey**</u>: Test whether a key is collected after an AI tank moves over it. AI should always follow the key when it is unoccupied.
Test items: Software code - checkForKey(), followKey(), movementControl().
Input specifications: the key is at a random position on the field and one tank moves over the key.
Output specifications: the key starts to move along with the tank.
Environmental needs: key has not been picked up and is on a random position, one ai tank passing over it.

<u>**AIFollowKeyHolder**</u>: Test whether the AI tank follows the key when it is picked up by a tank.
Test items: Software code - followKeyHolder(), movementControl().
Input specifications: one tank picks the key, the key moves along with it.
Output specifications: the AI tank starts to follow the key holding tank.
Environmental needs: one tank from the opposite team holding the key.

<u>**AIFollowObjective**</u>: Test whether an AI tank is striving to reach the objective once it is holding the key.
Test items: Software code - checkForObjective(), incrementTeamScores(), followObjective(), movementControl().
Input specifications: AI tank is currently holding the key.
Output specifications: the AI tank reaches the objective and stay there until the key is respawned.
Environmental needs: AI tank holding the key.

<u>**TankRespawn**</u>: Test whether a tank is being reset at its initial position on the screen after it has been killed.
Test items: Software code - resetTank(), hitTank().
Input specifications: one tank loses all his health points and is killed.
Output specifications: the tank appears at its initial position after five seconds.
Environmental needs: one tank with 0 health points.

<u>**AIShootAtOpponents**</u>: Test whether the AI tank shoots at the direction of the opponent tank.
Test items: Software code - startShooting(), shoot().
Input specifications: The AI tank is near an opponent tank.
Output specifications: The AI tank shoots at the direction of the opponent tank
Environmental needs: An AI tank and an opponent tank connected to the server and standing close by each other.

<u>**AIAvoidObstacles**</u>: Test whether the AI tank follows the key when it is picked up by a tank.
Test items: Software code - doSearch(), updatePath(), followPath(), movementControl()
Input specifications: The AI tank is placed in a map with obstacles

Output specifications: The AI tank tries to reach its goal while avoiding obstacles and finding the shortest path.
Environmental needs: An AI tank placed on a map, containing various obstacles.

<u>**AIRotateGun**</u>: Test whether an AI tank is rotating its gun towards the direction of shooting.
Test items: Software code - startShooting(), shoot(), calcTopAngle().
Input specifications: An AI tank and an opponent tank nearby
Output specifications: The AI gun should rotate pointing to the direction of shooting.
Environmental needs: An AI tank and an opponent tank connected to the server and standing close by each other.

<u>**AISearchShortestPath**</u>: Test whether the AI tank is finding the shortest path towards a goal.
Test items: Software code - doSearch(), updatePath(), followPath()
Input specifications: An AI tank and a goal to reach.
Output specifications: The AI tank should do a search through the map and follow the found shortest path.
Environmental needs: An AI tank placed in a map and given a goal to reach.

Because of the nature of our parser we have decided to adopt more extensive strategy for them. Therefore, we decided to test them with even more inputs than usual. Thus, we were able to ensure their functionality.

Test Case ID	Description	Inputs			Actual Outputs			Match expected output?
		1) Standard	2) Boundary	3.)Boundary	1)	2)	3)	
TC001	Lobby Messaging	"Hello"	"Hello!!Hello "	"#Hello"	"Hello" is sent to server and displayed for all clients	Parse symbols are handled, "Hello!!Hello" displayed for all clients	"#Hello" displayed for all clients	Yes
TC002	Message auto-scrolling	10 lines of input	0 lines of input	1 line of input	Scroll bar auto scrolls accordingly with most recent line	No scroll bar	No scroll bar	Yes
TC003	Set Nickname	"Keith"	""	"#!!Hi!!##"	Nickname set to Keith, chat and teams vary accordingly	Nickname rejected	Nickname set accordingly	Yes
TC004	Client Ready	client clicks ready			Server notified of ready, ready button disabled for client, the table is updated for all clients			Yes
TC005	Host Start Button	All clients ready	no clients ready	one client ready	Start game button enabled	Start game button disabled	Start game button disabled	Yes
TC006	Switch Team	Toggle to open team	Toggle to closed team		Player details change table, unready	Player Details change table, Excess player highlighted red, blocks game start, unready		Yes
TC007	Host IP	Lobby entered			Host WLAN IP displayed			No
TC008	Leave Game	Leave Button pressed			Connection closed, game unreadied, Teams updated			Yes
TC009	Start Game	Button Pressed			Game threads begin			Yes

TC010	Client joins multiplayer session	Valid IP of host	""	"hfdsjk"	Joins session and client is instantiated with lobby history	Attempts to join localhost	Unable to connect	Yes
TC011	User starts single player game	Button pressed			Player enters lobby with single AI			Yes
TC012	User creates multiplayer session	Button pressed			Player enters lobby with appropriate AI, and waits for players to join			Yes

Description:
Tests 001 - 010 are lobby tests which ensure pre-game state functionality is behaving properly. They check that users can properly communicate in order to start the game.
Tests 011 - 013 Test the integration of the networking and the GUI to ensure that the clients connect properly and that the GUI changes panel.

3. USABILITY TESTS

As a usability testing strategy we did a presentation of our game in front of some of our peers. We let them play and we asked for feedback. Also, we were prepared with specific questions reflecting on the things that we thought were most important for our system

- Learnability – how fast and intuitive our game appears.
- Robustness – are the players fully aware of all achievements and benefits in terms of gameplay.
- Flexibility – is the user-system communication fast and comprehensive to ensure pleasant game experience.

Based on the information that we have gathered we did the following changes:

- We added a keyhole in the middle of our objective to make the main objective more intuitive.
- We reduced the time needed for the key to be hold on the objective because some users found it too difficult the way it was.
- We added a range to the bullets so we decrease the difficulty as pointed out.
- We added visual aids for tracking the scores, the time left to get key points and an end screen showing nicely whether you have won or not.
- Some users pointed out that the lagging was quite annoying so we reduced the frames shown per second and thus we achieved smoother gameplay.

4. COLOUR-BLIND TESTS

To ensure that the colours we used in the game where friendly to colour-blind players we used the vision simulator from www.vischeck.com/vischeck/vischeckImage.php. This simulator allowed us to simulate the following colour visions:

- Deuteranope (a form of red/green colour deficit)
- Protanope (another form of red/green colour deficit)
- Tritanope (a blue/yellow deficit – very rare)

After a number of tests we decided to use the following colours because they were the most distinguishable colours and they didn't have a big impact on the overall quality of the game:

Health colours

- Blue for allies
- Red for enemies
- Yellow for own tank

Tanks

- Blue for Team 1
- Red for Team 2

Timer

- Black
- Blue when team 1 is winning
- Red when team 2 in winning

Below is some the test we ran, before and after the changes:

4.1. Before Changes

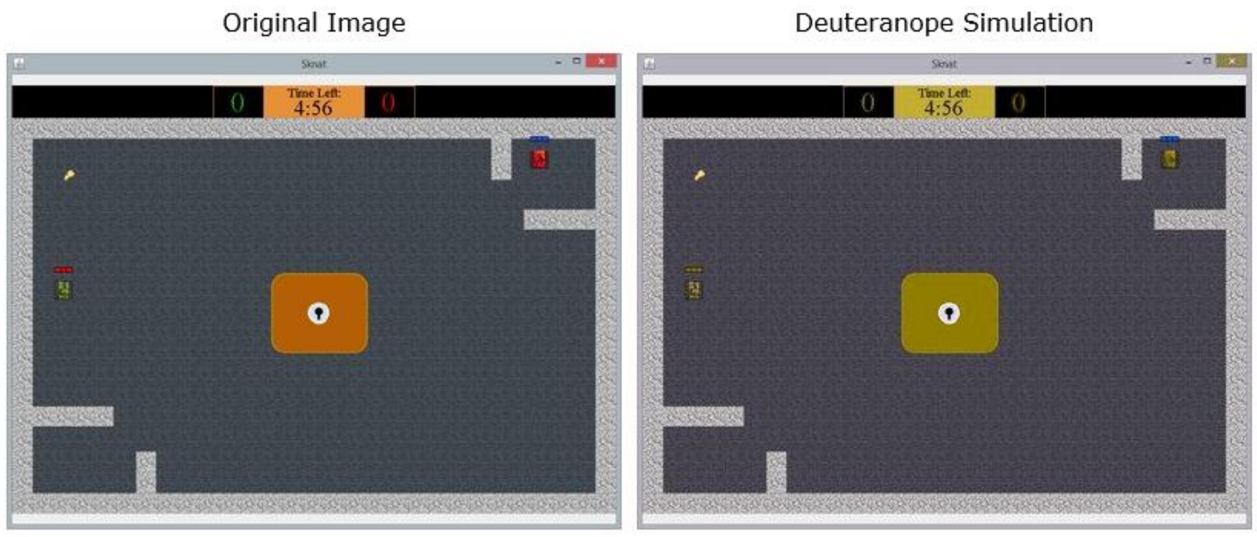


Figure 1: In this test the tanks look like they have the same colour.

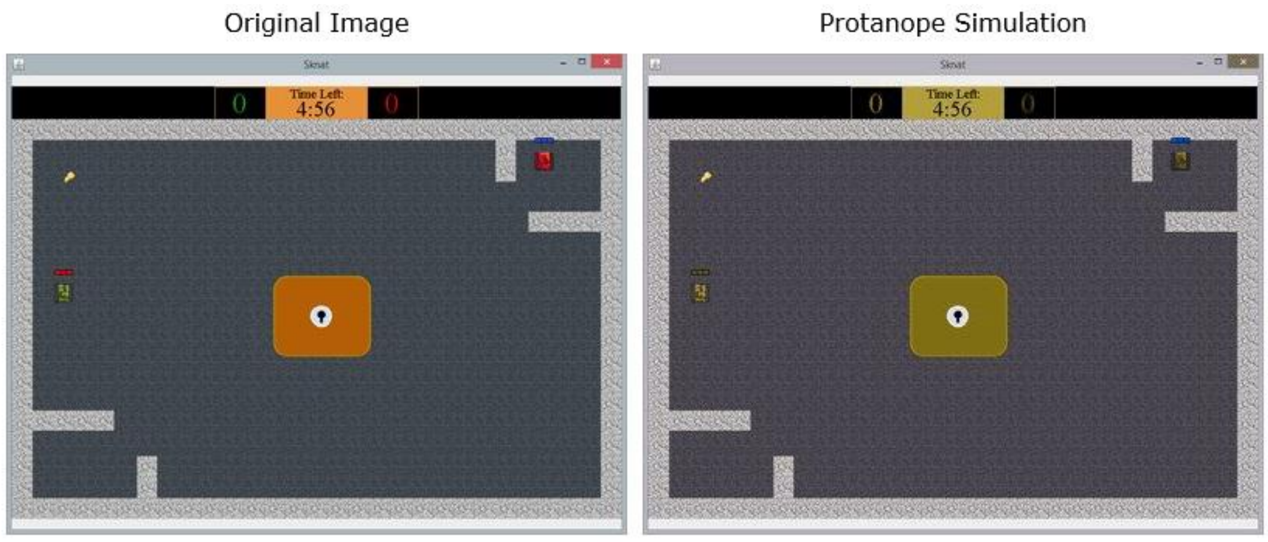


Figure 2: In this case the tanks look like they have the same colour but different shades, which is not too bad, but will definitely have an impact on the gameplay.

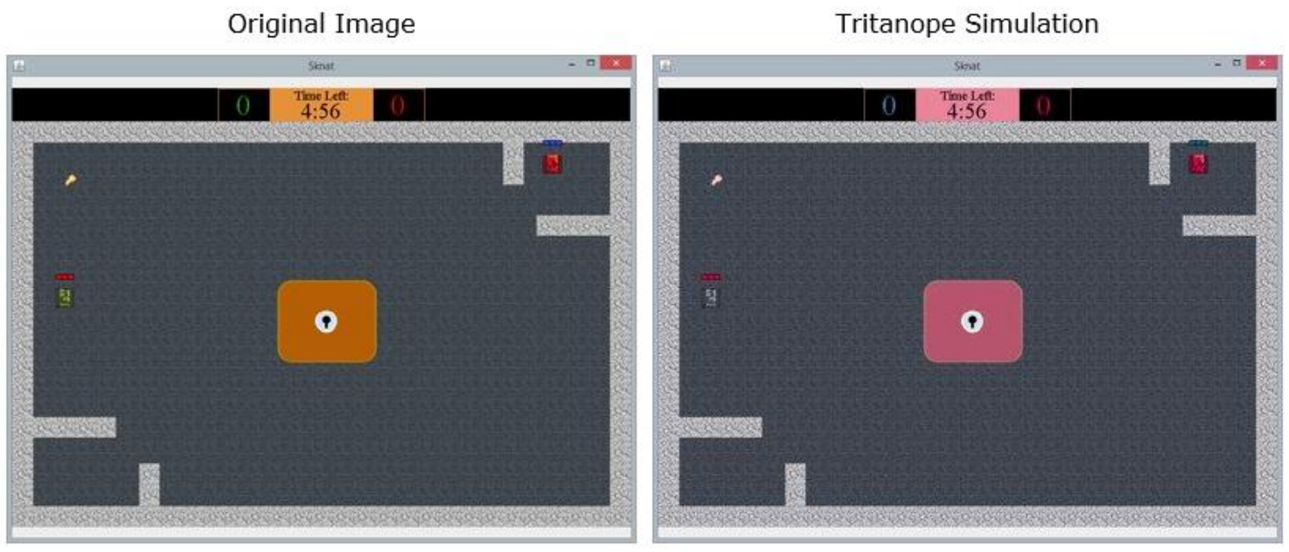
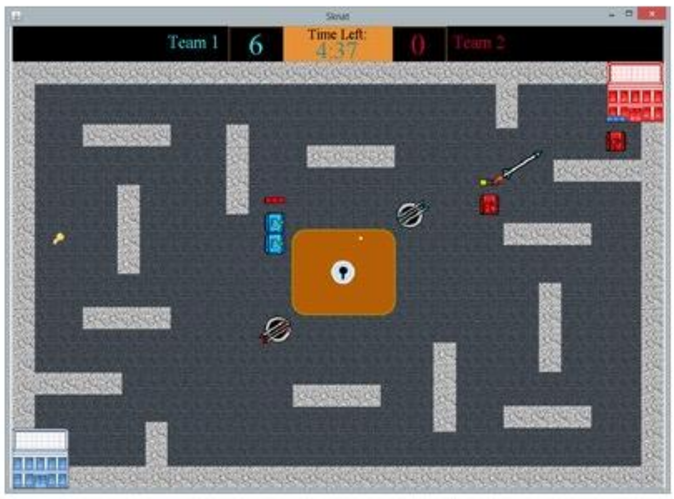


Figure 3: In this simulation the choice of colours does not really have an impact on the gameplay because all the colours are distinguishable, even though it looks different from the original game.

4.2. After Changes

Original Image



Deuteranope Simulation

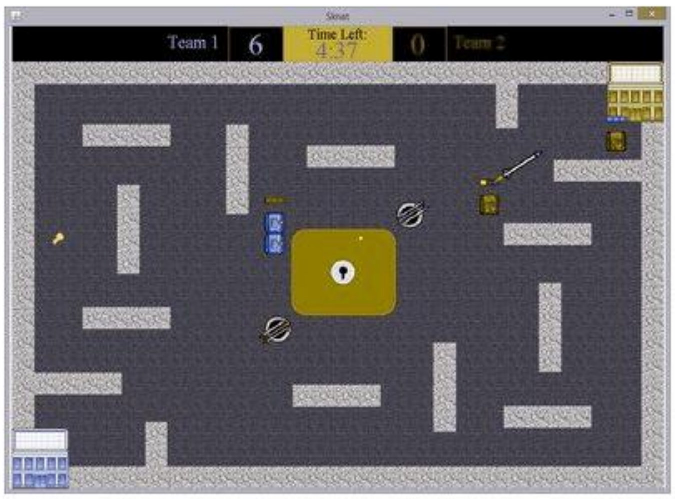
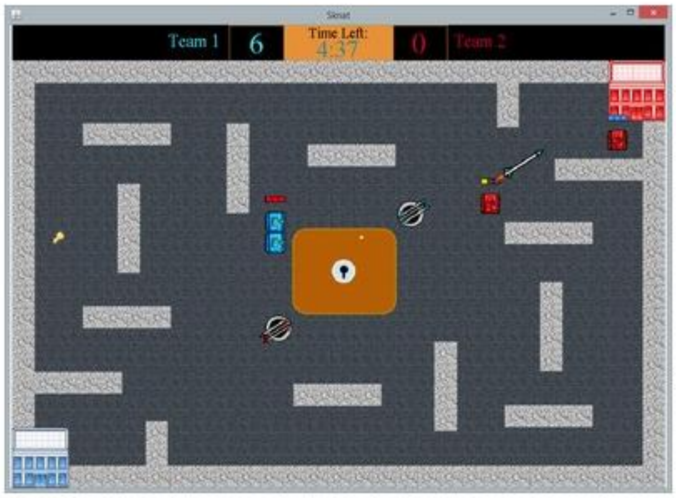


Figure 4: After changes the tanks do not look the same anymore in this simulation.

Original Image



Protanope Simulation

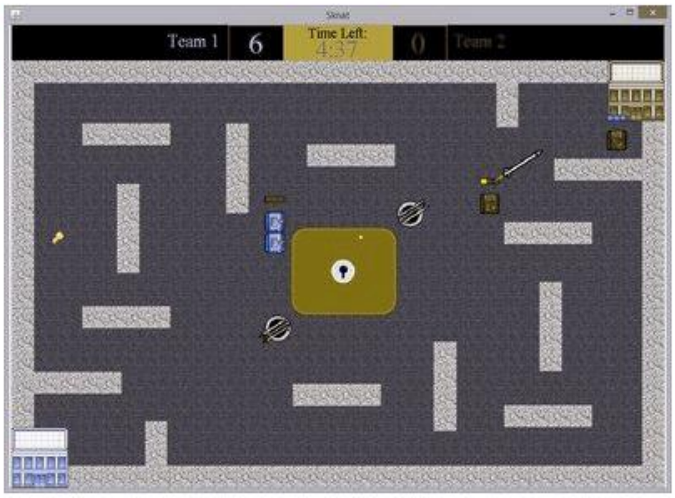
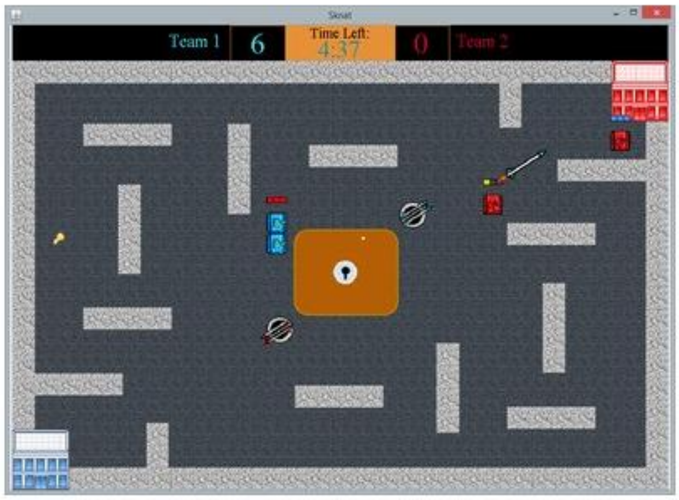


Figure 5: The use of blue makes it very easy to identify and distinguish tanks on the battlefield.

Original Image



Tritanope Simulation

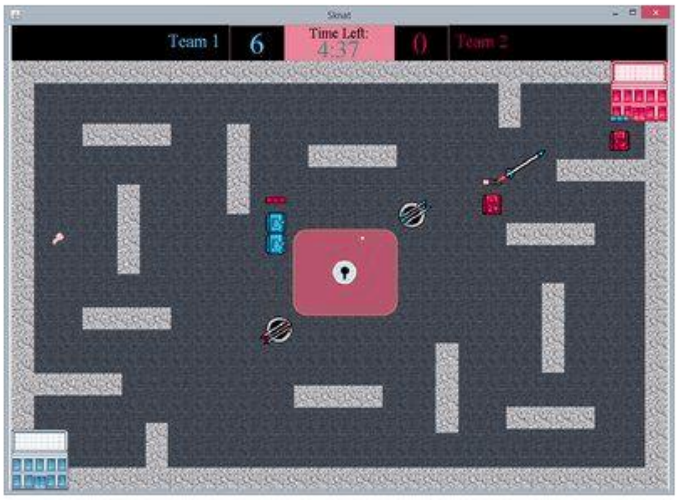


Figure 6: This simulations shows how this type of colour deficiency is not impacted that much just like before the changes.

5. Regression Testing

We used regression testing during the development of the game. When adding extensions or making any changes to the game, we made sure that all the existing JUnit tests and functional test were correct. This allowed us to have a stable version of the game even in development stages.