CS 111 Final Project  
Self Assessment

Your group should fill out one copy of this form and include it with your assignment when you turn it in.

# Group

*Who’s in your group?*

1. Person1: Erick Mungai
2. Person2: Abel Ariko
3. Person3: Vernon Otieno

# Goals

*Say a few words about what you wanted the game to be like. Note that if you just wanted to write some code so you could get a good grade on the project, it’s fine to admit that.*

We wanted the game to not only achieve the objectives spelt out it the final project description, but also interesting to the player. We therefore came up with an interesting synopsis for the game where a hunter stumbled into a cabin in the woods, not knowing it belonged to a witch, and was trapped inside. He had to find a key hidden in the second room so that he could escape. However, a bomb was also hidden in the possible places where the key could have been hidden. Finding the bomb would defuse it and the hunter would die causing the game to end. If the hunter found the key, he could proceed to the next room where he would find a horse to escape with.

# Lessons learned

*What went right?*

Making the different objects was easy. Writing their methods was also not complicated.

In the end, we were able to accomplish building the game as we had set out to do and it functioned fluidly.

*What went wrong?*

We wanted to implement a timer, where the player would have a specified timeframe during which they could find the key. If time ran out, the witch would come back and cast a spell that would make the hunter die thus ending the game. However, this was fairly difficult, and we abandoned the idea.

*What do you wish you knew when you started?*

If we knew about creating timers in Racket, it would have greatly helped us and we wouldn’t have wasted so much time trying to implement the idea.

Also, if we knew about how to use (the …) appropriately.

This is because we couldn’t use it to access objects that were in other objects in a room. For example, we put food inside a cabinet but we couldn’t use (the food). We first had to take the food so that it was now in the room container and access it using (the)

# Annoying grading bookkeeping

## Types

*What are the types you added, and what are they for?*

1. Type1: Horse- For creating horse objects. A horse would assist the player to run away from the cabin when he got to the final room.
2. Type2: Bomb- For creating bomb. Finding a bomb makes the player explode and die thus ending the game.
3. Type3: Key- For creating keys for accessing locked rooms.
4. Type4: Food- For creating food objects. They are meant to be eaten by the player
5. Type5: Gramophone- Creates gramophones that can be switched on to play music and off to stop playing music.
6. Type6: Rocking chair- Creates chairs that the player can sit on.
7. Type7: Cabinet- Creates cabinets where other objects can be stored i.e food is stored inside the cabinet
8. Type8: Gun- Creates guns

## Fields

*What are the fields you added, what types did you add them to, and what are they for?*

1. Field1: hidden-location->Bomb. Stores the location where the bomb is hidden
2. Field2: hidden-location->Key. Stores the location where the key is hidden.
3. Field3: on? ->Gramophone. Boolean for whether the gramophone is on or off.
4. Field4: seated-down? ->Rocking chair. Boolean for whether the player is sitting on the chair or not.
5. Field5: leaned-back? ->Rocking chair. Boolean for whether the chair is leaned back or not.
6. Field6: opened? ->Cabinet. Boolean for whether the cabinet is open or not. You can’t take objects out of the cabinet without opening it first.

We also added a field to door

1. Field7: locked? -> Door. Boolean for whether a door is locked or not.
2. Field8: loaded? -> Gun. Boolean for whether a gun is loaded or not.

## Procedures

*What are the procedures you added or significantly modified from their original form, and what are they for?*

1. Procedure1: check-under: enables the user to check where a key is hidden
2. Procedure2: new-horse -> initializes new horses
3. Procedure3: new-food-> initializes new food
4. Procedure4: new-gramophone-> initializes new gramophones
5. Procedure5: new-key-> initializes new keys
6. Procedure6: new-bomb-> initializes new bombs
7. Procedure7: new-cabinet-> initializes new cabinets
8. Procedure8: new-rocking-chair-> initializes new rocking-chairs
9. Procedure9: new-gun-> initializes new guns

## Methods

*What are the methods you added or significantly modified from their original form, what types were they added to, and what are they for? Note that if you have three different methods for the same generic procedure, list each one separately.*

1. Method1: ride ->horse: enables the player to ride a horse and escape
2. Method2: eat-> food: enables the player to eat food
3. Method3: switch-on -> gramophone: enables the player to switch on the gramophone
4. Method4: switch-off -> gramophone: enables the player to switch off the gramophone
5. Method5: sit-down-> rocking-chair: enables the player to sit down on the chair
6. Method6: stand-up-> rocking-chair: enables the player to stand up if he’s sitting down
7. Method7: lean-back ->rocking-chair: enables the player to lean back the chair
8. Method8: lean-up ->rocking-chair: enables the player to make the chair lean up
9. Method9: open -> cabinet: enables the player to open the cabinet
10. Method10: close-> cabinet: enables the player to close the cabinet
11. Method11: load-> gun: enables the player to load the gun

## Total stuff we built

*Write the total number of items listed above.*

1. horse
2. bomb
3. key
4. food
5. gramophone
6. rocking-chair
7. cabinet
8. gun
9. check-under
10. new-horse
11. new-food
12. new-gramophone
13. new-key
14. new-bomb
15. new-cabinet
16. new-gun
17. new-rocking-chair
18. hidden-location(bomb)
19. hidden-location(key)
20. on?
21. seated-down?
22. leaned-back?
23. opened?
24. loaded?
25. locked
26. switch-on
27. switch-off
28. sit-down
29. stand-up
30. lean-back
31. lean-up
32. open
33. close
34. ride
35. eat
36. load

*Score we deserve:*

100%

Apart from meeting the bare minimum required by the instructor, the creators of this game went above and beyond to create an interesting and captivating experience for the user. Drawing inspiration from shows such as “Into the Badlands and “Escape room”, the creators were able to come up with a game that has suspense, action and adventure.