Project:Terminal Application

An adventure game that started out with huge ambitions that never came to be

未来トイレ

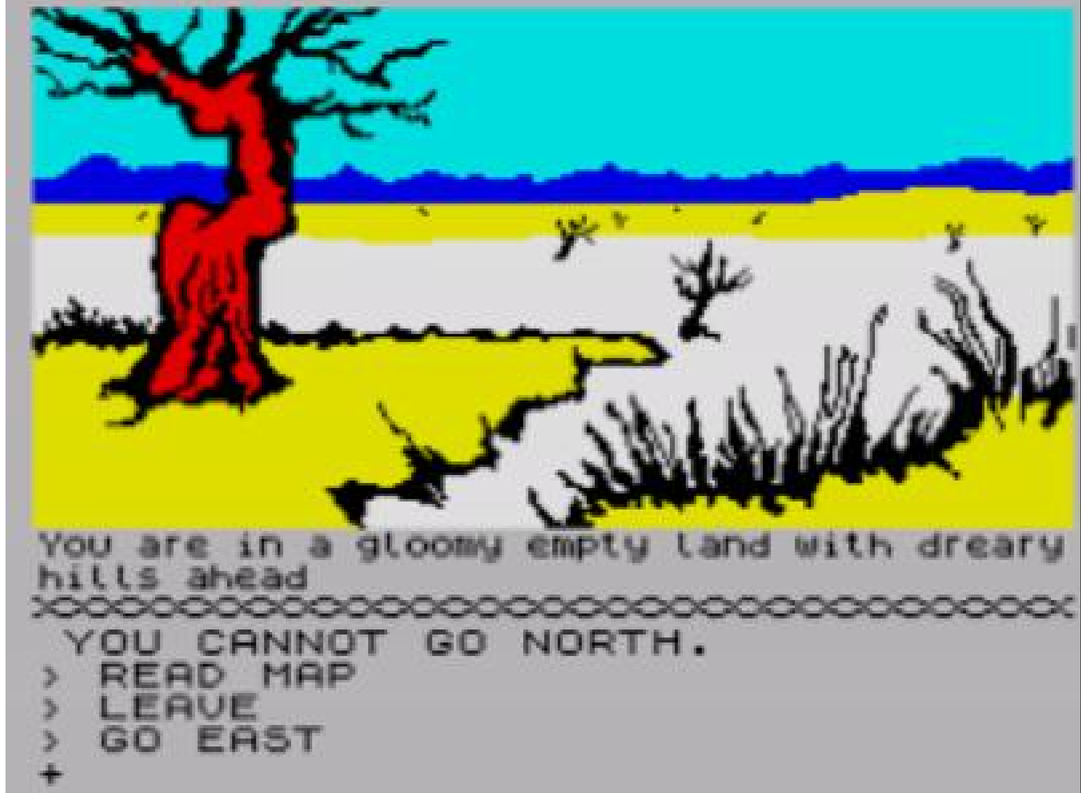
Mirai Toire

Mirai Toire GO

未来 - Future

トイレ - Toilet

Adventure games are hard...



You are in a gloomy empty land with dreary hills ahead

```
 YOU CANNOT GO NORTH.
> READ MAP
> LEAVE
> GO EAST
+
```

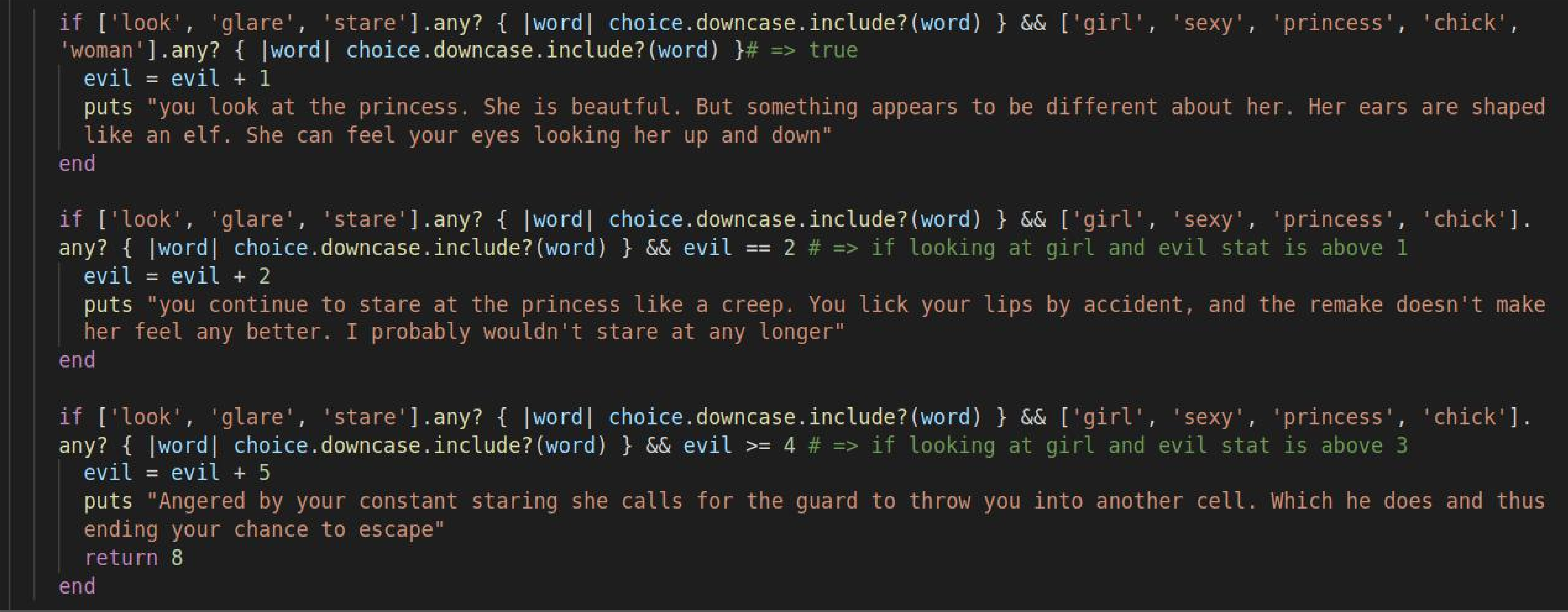
Originally....

My initial goal was to create an experience like no other. The idea was that you were someone waking up in a room, unsure of how you arrived there. The only thing available to you is a weird looking toilet. Sitting on the toilet and leaving a desposit sample would allow you to travel anywhere in any time or place.

That meant that you could travel to West Germany during World War 2. Or travel and meet The Beatles or Jimi Hendrix at Woodstock. But it wasn't mearly travelling to those locations, but you could type anything and a fairly correct response would be returned.

None of this 'can't understand this command'.

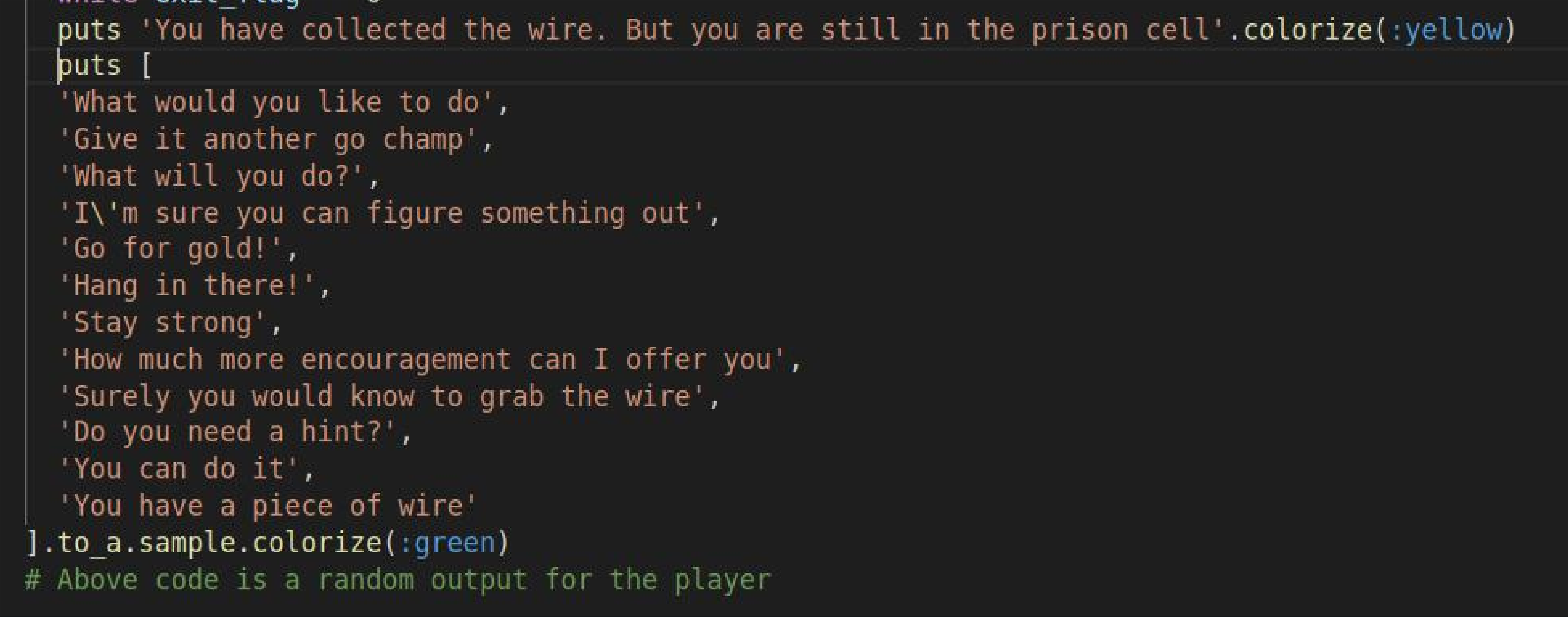
The reason for this was a test for the human condition, and allow people to do anthing that might seem uncouth. Such as going back and kidnapping baby Jesus.



```
if ['look', 'glare', 'stare'].any? { |word| choice.downcase.include?(word) } && ['girl', 'sexy', 'princess', 'chick',
'woman'].any? { |word| choice.downcase.include?(word) }# => true
  evil = evil + 1
  puts "you look at the princess. She is beautful. But something appears to be different about her. Her ears are shaped
  like an elf. She can feel your eyes looking her up and down"
end

if ['look', 'glare', 'stare'].any? { |word| choice.downcase.include?(word) } && ['girl', 'sexy', 'princess', 'chick'].
any? { |word| choice.downcase.include?(word) } && evil == 2 # => if looking at girl and evil stat is above 1
  evil = evil + 2
  puts "you continue to stare at the princess like a creep. You lick your lips by accident, and the remake doesn't make
  her feel any better. I probably wouldn't stare at any longer"
end

if ['look', 'glare', 'stare'].any? { |word| choice.downcase.include?(word) } && ['girl', 'sexy', 'princess', 'chick'].
any? { |word| choice.downcase.include?(word) } && evil >= 4 # => if looking at girl and evil stat is above 3
  evil = evil + 5
  puts "Angered by your constant staring she calls for the guard to throw you into another cell. Which he does and thus
  ending your chance to escape"
  return 8
end
```

if statements

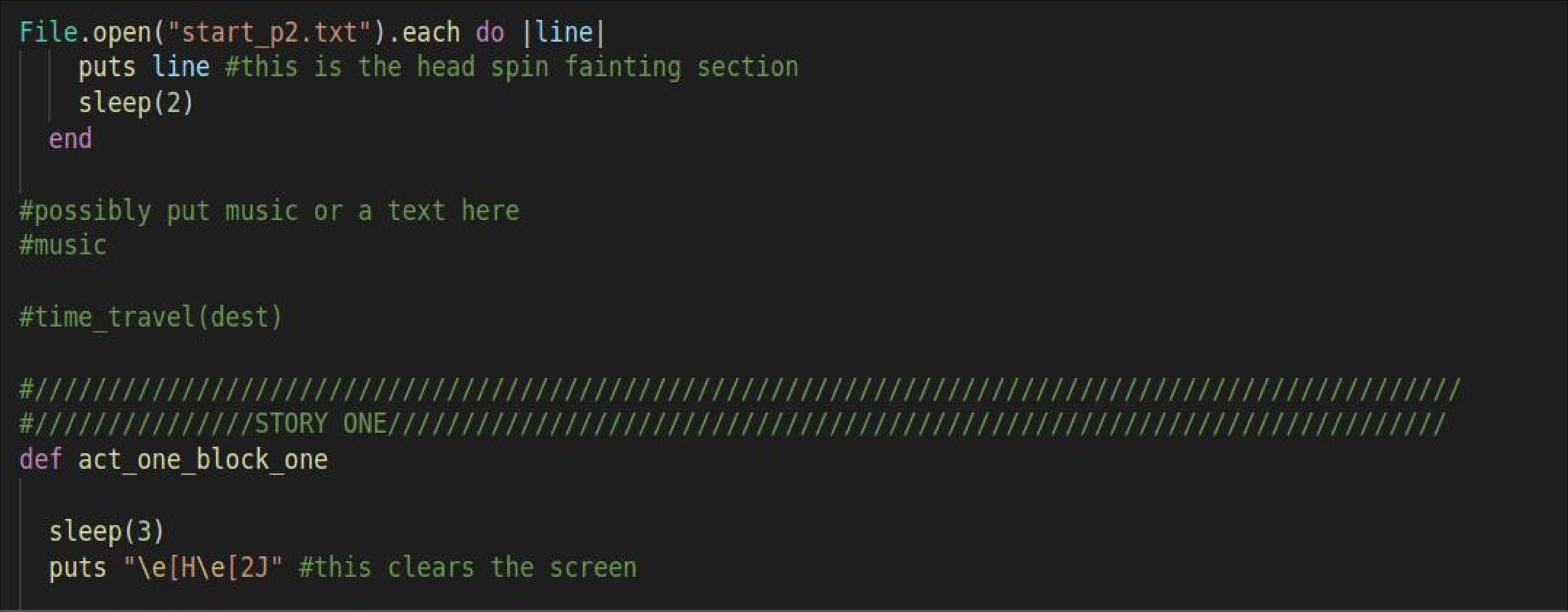
- Most of the heavy lifting is done with 'if' conditions. When the user enters a statement it will check whether the word matches in the array.
- the downcase allows the user to enter both uppercase and lowercase and the condition will still be true.
- In this example, I have also included a stat system depending on what the player enters.



```ruby
  puts 'You have collected the wire. But you are still in the prison cell'.colorize(:yellow)
  puts [
  'What would you like to do',
  'Give it another go champ',
  'What will you do?',
  'I\'m sure you can figure something out',
  'Go for gold!',
  'Hang in there!',
  'Stay strong',
  'How much more encouragement can I offer you',
  'Surely you would know to grab the wire',
  'Do you need a hint?',
  'You can do it',
  'You have a piece of wire'
].to_a.sample.colorize(:green)
# Above code is a random output for the player
```

random statements

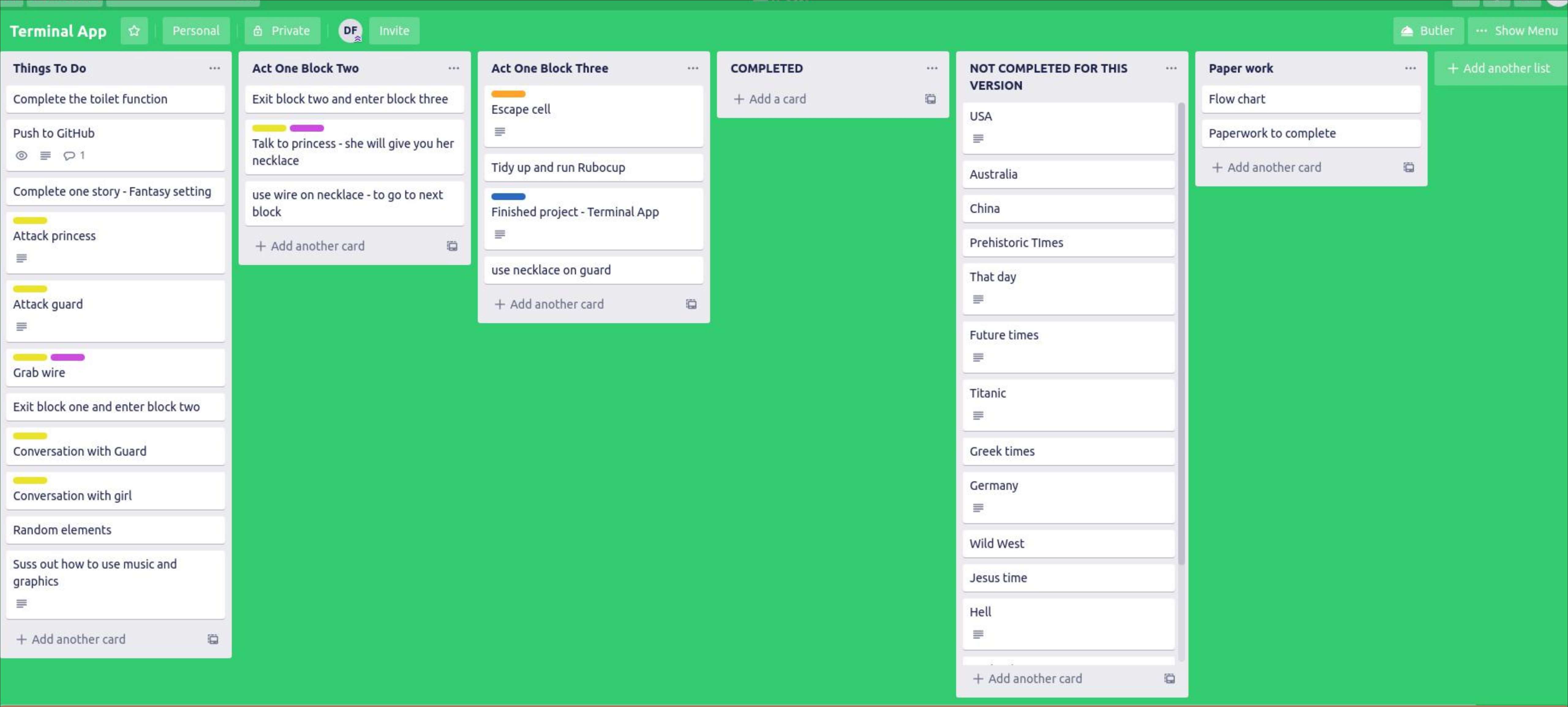
- Some of the code features random features such as dialogue.
- This example shows 'sample' being called on an array, which randomise the output
- Another feature was never ended up was the rand(i) function that can output a random number ie rand(5) => random number from 0 to 5



```ruby
File.open("start_p2.txt").each do |line|
    puts line #this is the head spin fainting section
    sleep(2)
  end

#possibly put music or a text here
#music

#time_travel(dest)

#/////////////////////////////////////////////////////////////////////////////////////////////////
#///////////////STORY ONE////////////////////////////////////////////////////////////////////////
def act_one_block_one

  sleep(3)
  puts "\e[H\e[2J" #this clears the screen
```

importing of external docs

- Importing text files and using them instead of type out lines of code. Would also help with creating ascii art in a text document
- sleep(3) is a neat feature that can helps the flow of the program
- puts “\e[H\e[2J” clears the screen. Yep, had to google that one


Terminal App

Things To Do
- Complete the toilet function
- Push to GitHub
- Complete one story - Fantasy setting
- Attack princess
- Attack guard
- Grab wire
- Exit block one and enter block two
- Conversation with Guard
- Conversation with girl
- Random elements
- Suss out how to use music and graphics

Act One Block Two
- Exit block two and enter block three
- Talk to princess - she will give you her necklace
- use wire on necklace - to go to next block

Act One Block Three
- Escape cell
- Tidy up and run Rubocup
- Finished project - Terminal App
- use necklace on guard

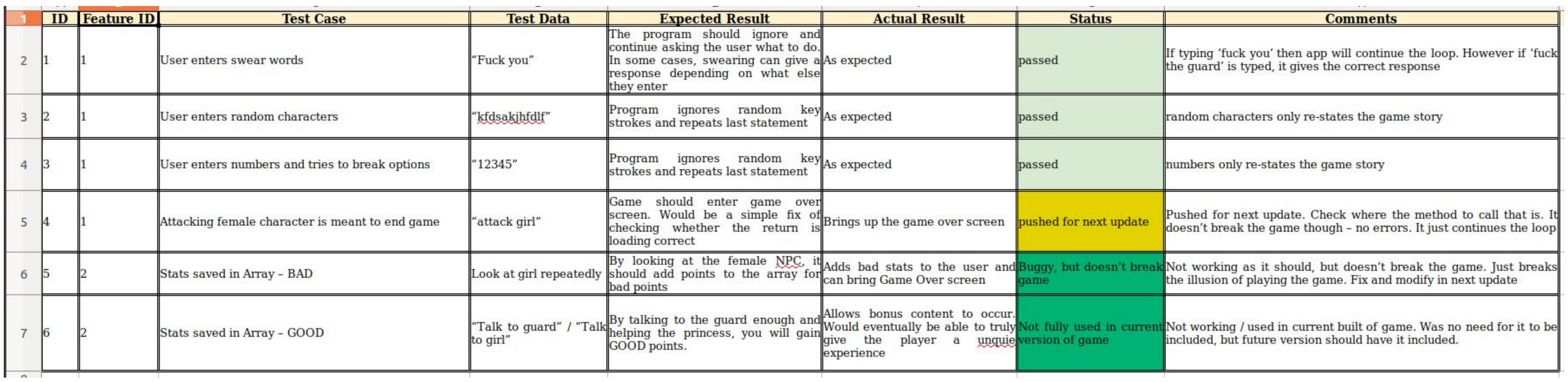
COMPLETED

NOT COMPLETED FOR THIS VERSION
- USA
- Australia
- China
- Prehistoric TImes
- That day
- Future times
- Titanic
- Greek times
- Germany
- Wild West
- Jesus time
- Hell

Paper work
- Flow chart
- Paperwork to complete

Planning Project Managment

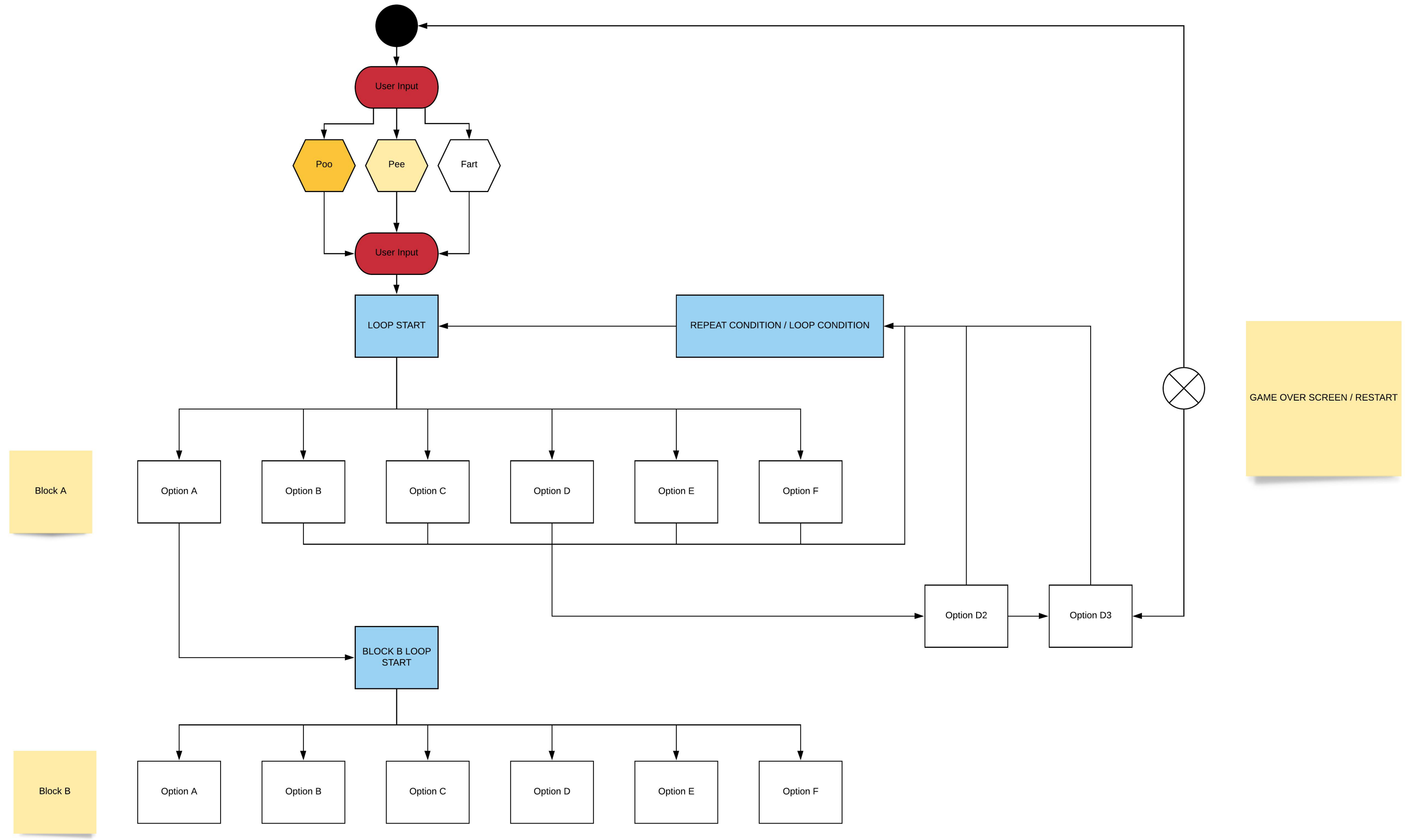
- Planning and project managment helped a lot to work out what I needed to complete.
- I found Trello too limited for what I wanted to do though, as it would be nice to have more options in setting flags. Though more time in the program would be needed



1	ID	Feature ID	Test Case	Test Data	Expected Result	Actual Result	Status	Comments
2	1	1	User enters swear words	"Fuck you"	The program should ignore and continue asking the user what to do. In some cases, swearing can give a response depending on what else they enter	As expected	passed	If typing 'fuck you' then app will continue the loop. However if 'fuck the guard' is typed, it gives the correct response
3	2	1	User enters random characters	"kfdsakjhfdlf"	Program ignores random key strokes and repeats last statement	As expected	passed	random characters only re-states the game story
4	3	1	User enters numbers and tries to break options	"12345"	Program ignores random key strokes and repeats last statement	As expected	passed	numbers only re-states the game story
5	4	1	Attacking female character is meant to end game	"attack girl"	Game should enter game over screen. Would be a simple fix of checking whether the return is loading correct	Brings up the game over screen	pushed for next update	Pushed for next update. Check where the method to call that is. It doesn't break the game though – no errors. It just continues the loop
6	5	2	Stats saved in Array – BAD	Look at girl repeatedly	By looking at the female NPC, it should add points to the array for bad points	Adds bad stats to the user and can bring Game Over screen	Buggy, but doesn't break game	Not working as it should, but doesn't break the game. Just breaks the illusion of playing the game. Fix and modify in next update
7	6	2	Stats saved in Array – GOOD	"Talk to guard" / "Talk to girl"	By talking to the guard enough and helping the princess, you will gain GOOD points.	Allows bonus content to occur. Would eventually be able to truly give the player a unquie experience	Not fully used in current version of game	Not working / used in current built of game. Was no need for it to be included, but future version should have it included.

Manual Error Checking

- Manual error checking



UML

- UML / Flow Chart of the process / control flow.
- In the program there are blocks. Like mini levels within the one level. As the player chooses the correct option it loads in the next block. If not, then the loop continues until the 'exit-flag' has been selected

Thanks. Hope to see you in the future..

(or the past.)