

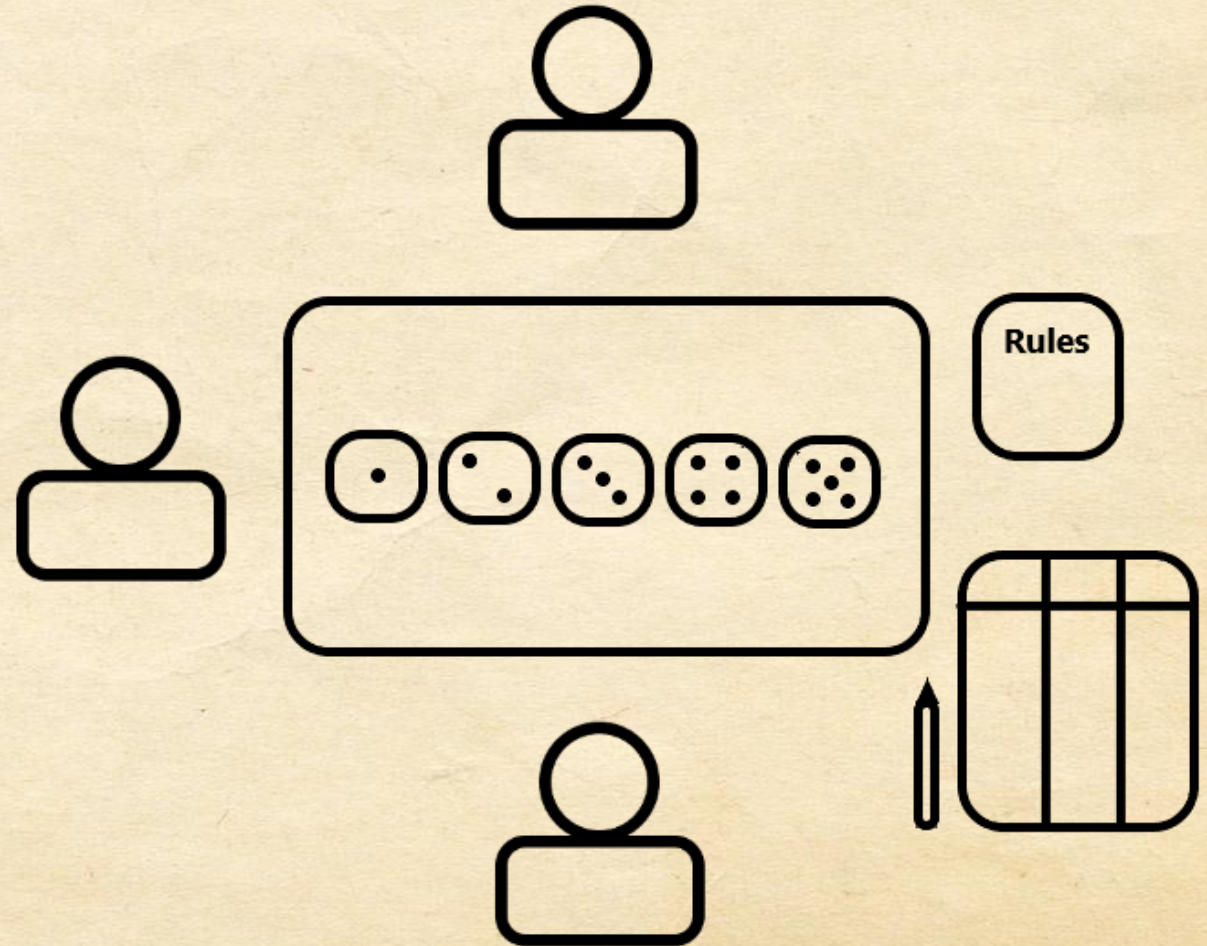


Thief Game Terminal App

What I planned/ what I learned

Inpiration: Thief, a fun hot dice game Invented by Cassandra Downs

- Played with 3-6 people
- Hot dice game like Yatzee
- Players earn points by rolling 1, 5 or a set > 2
- Players can steal points from each other at the risk of losing their turn
- First to 10,000 points wins!



Mechanics of the game

- A series of rounds are played until a winner is claimed
- Rounds start with the next active player who rolls 5 dice
- The result is either all valid, some valid or no valid
- If all valid the total chains and the player rolls 5 again
- If some valid the player has a choice
 - Hold all/some of valid dice and roll the remaining or
 - Pass and the next player can steal the turn and take over as active player
- Round ends when active player rolls no valid values or passes and their turn is not stolen



As a user... (early features planned)

- Access and run application through tty-prompt menu
- Learn the basic rules through a tutorial book
- Create a new game with custom player list
- Play through individual turns
- Be presented with game options
- Next player having option to steal
- Current player must be able to hold roll values

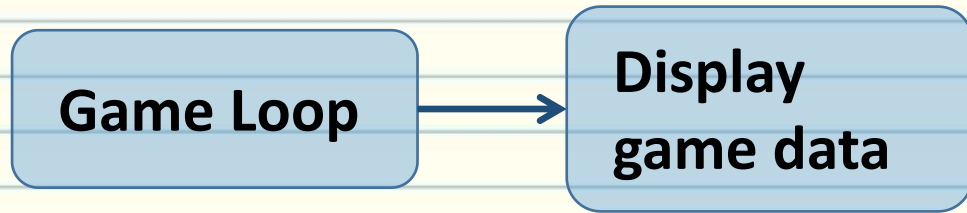
Welcome to my Theif Terminal Game!
Please select from the following: (Press
↑/↓ arrow to move and Enter to select)

- **Tutorial**
- **New Game**
- **Exit game**

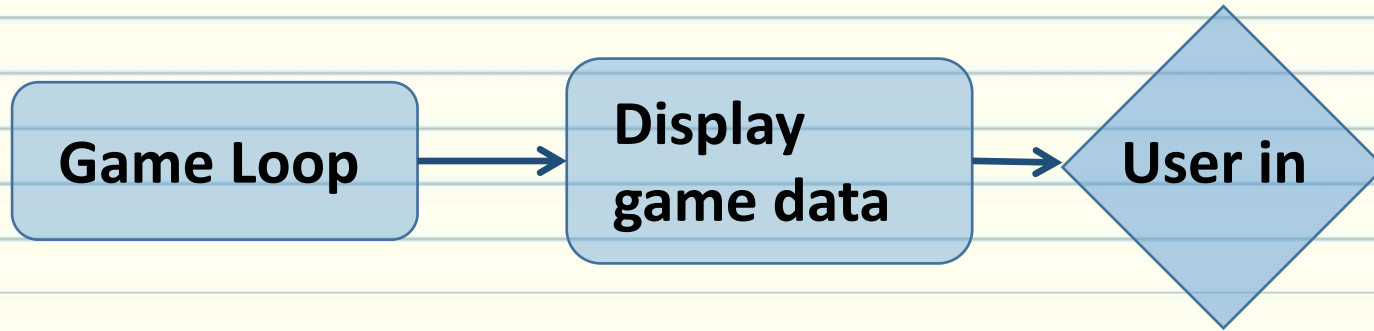
Control Flow: starting to make sense

Game Loop

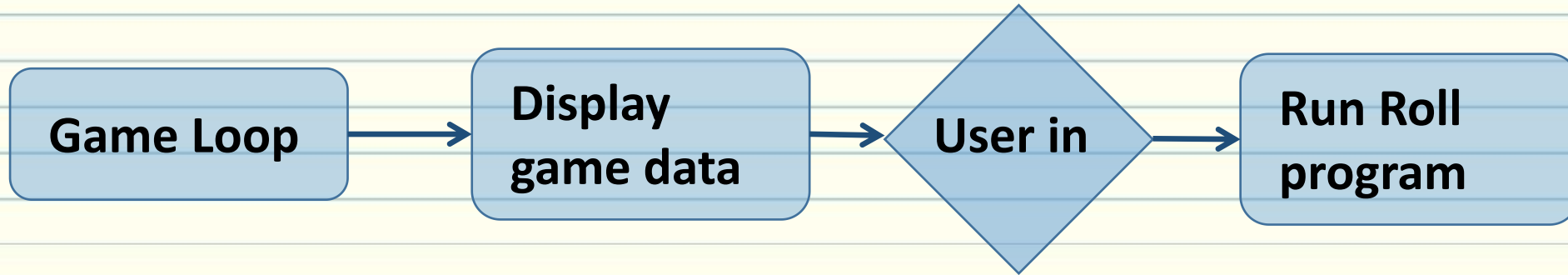
Control Flow: starting to make sense



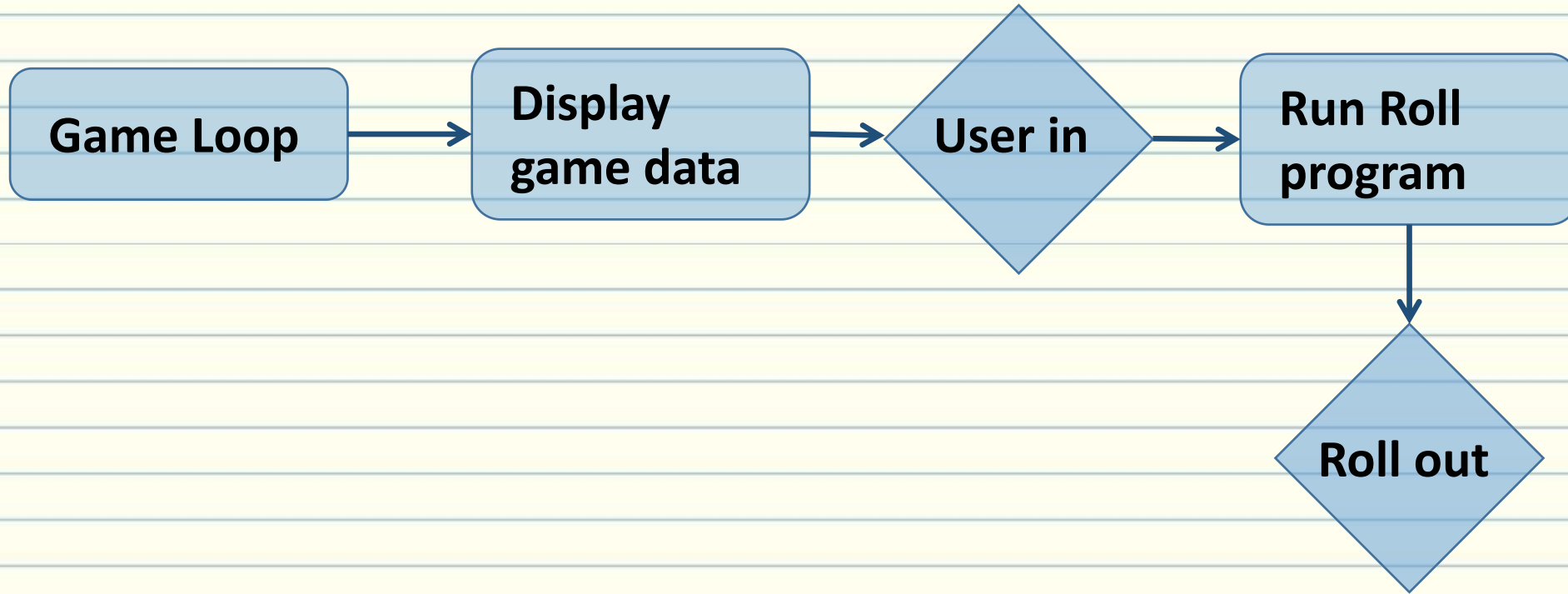
Control Flow: starting to make sense



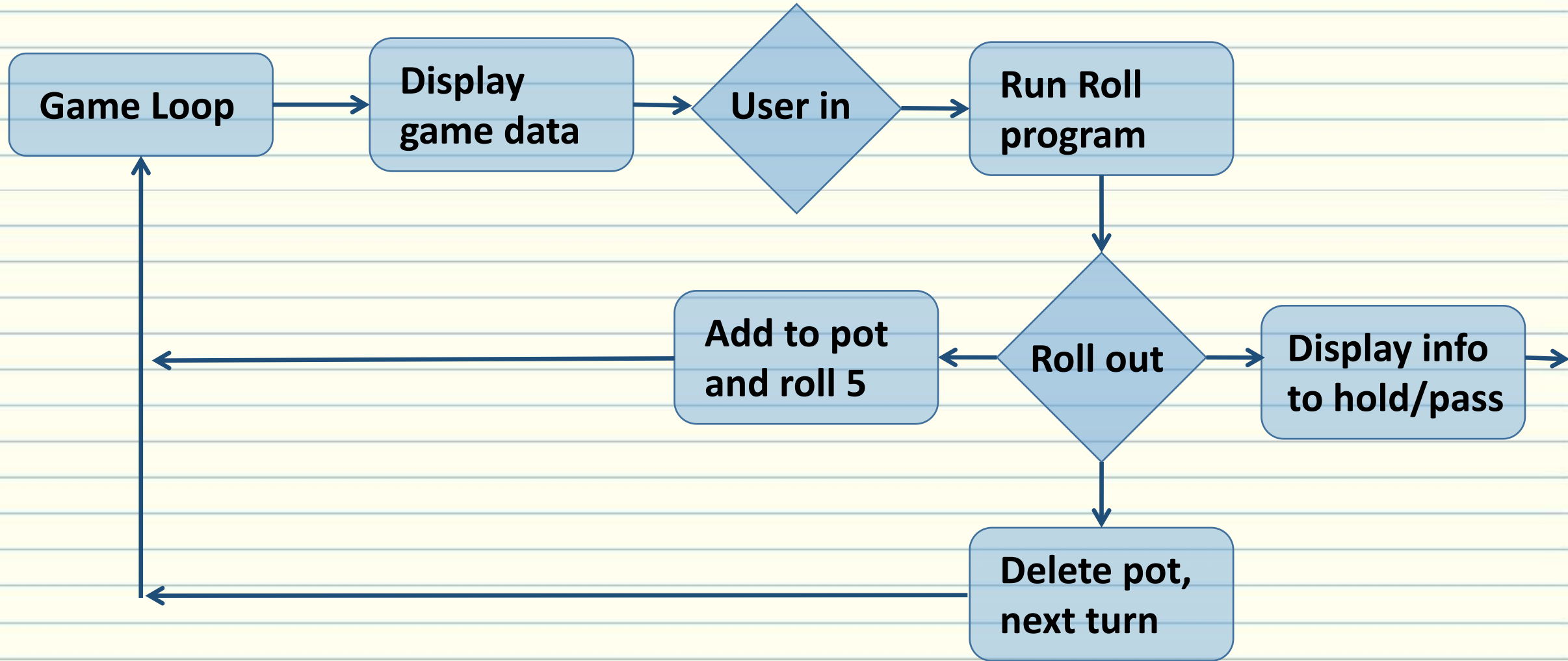
Control Flow: starting to make sense



Control Flow: starting to make sense

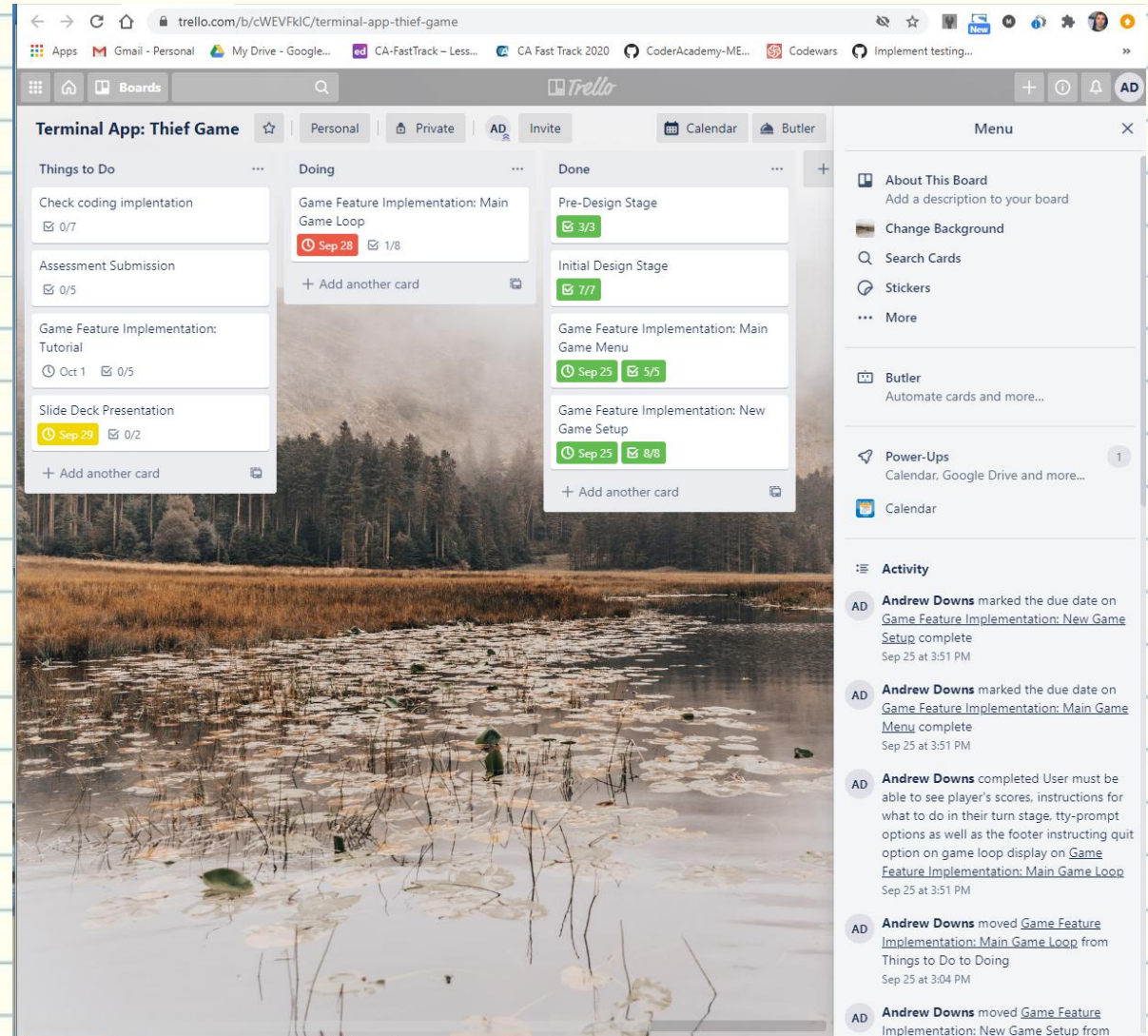


Control Flow: starting to make sense



Implementation: Solid start, so promising...

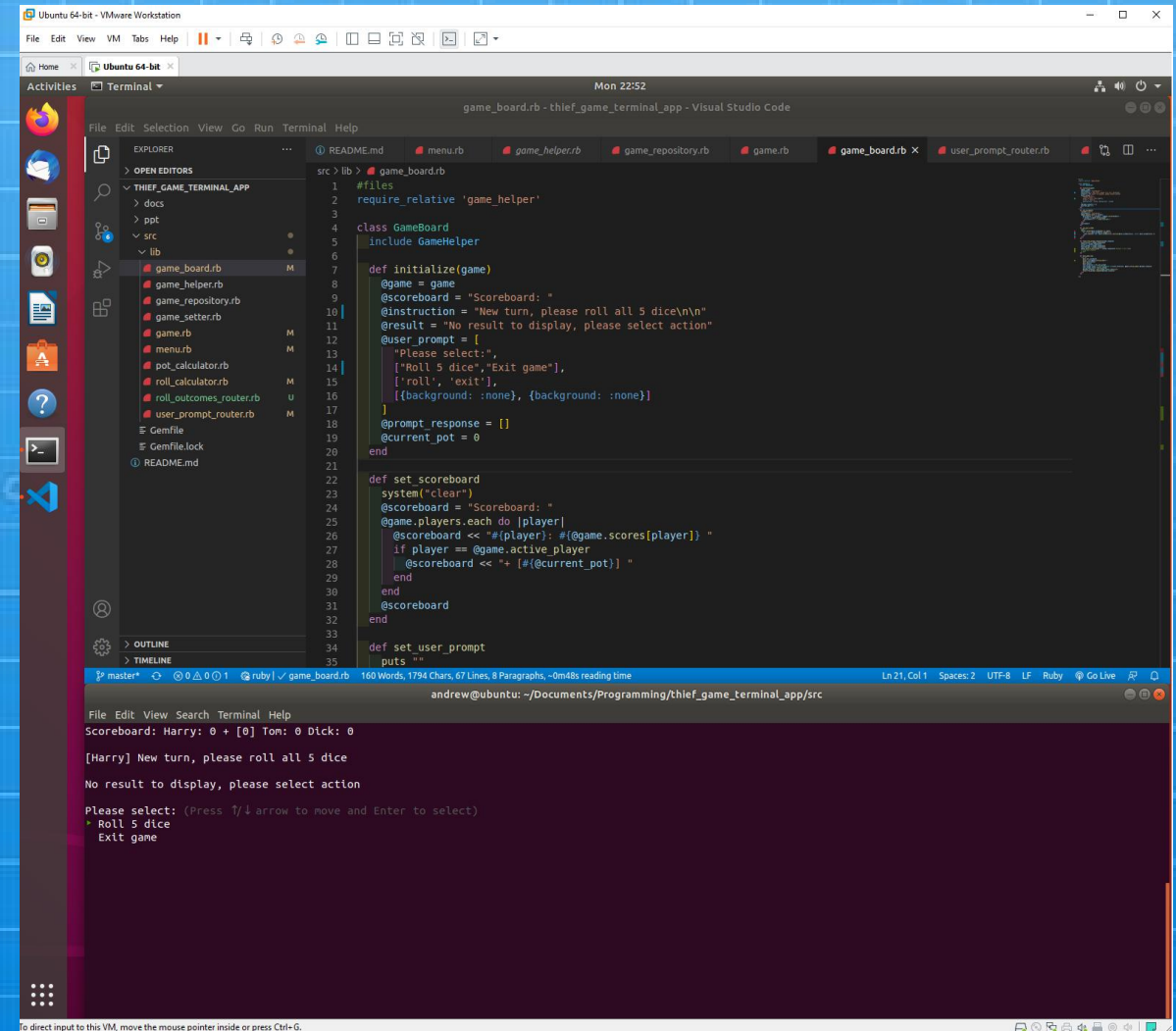
- Initial Design phase (check)
- Game start menu (check)
- Game setup menu (check)
- Main game Loop



Game logic can have so many classes!

Learning points:

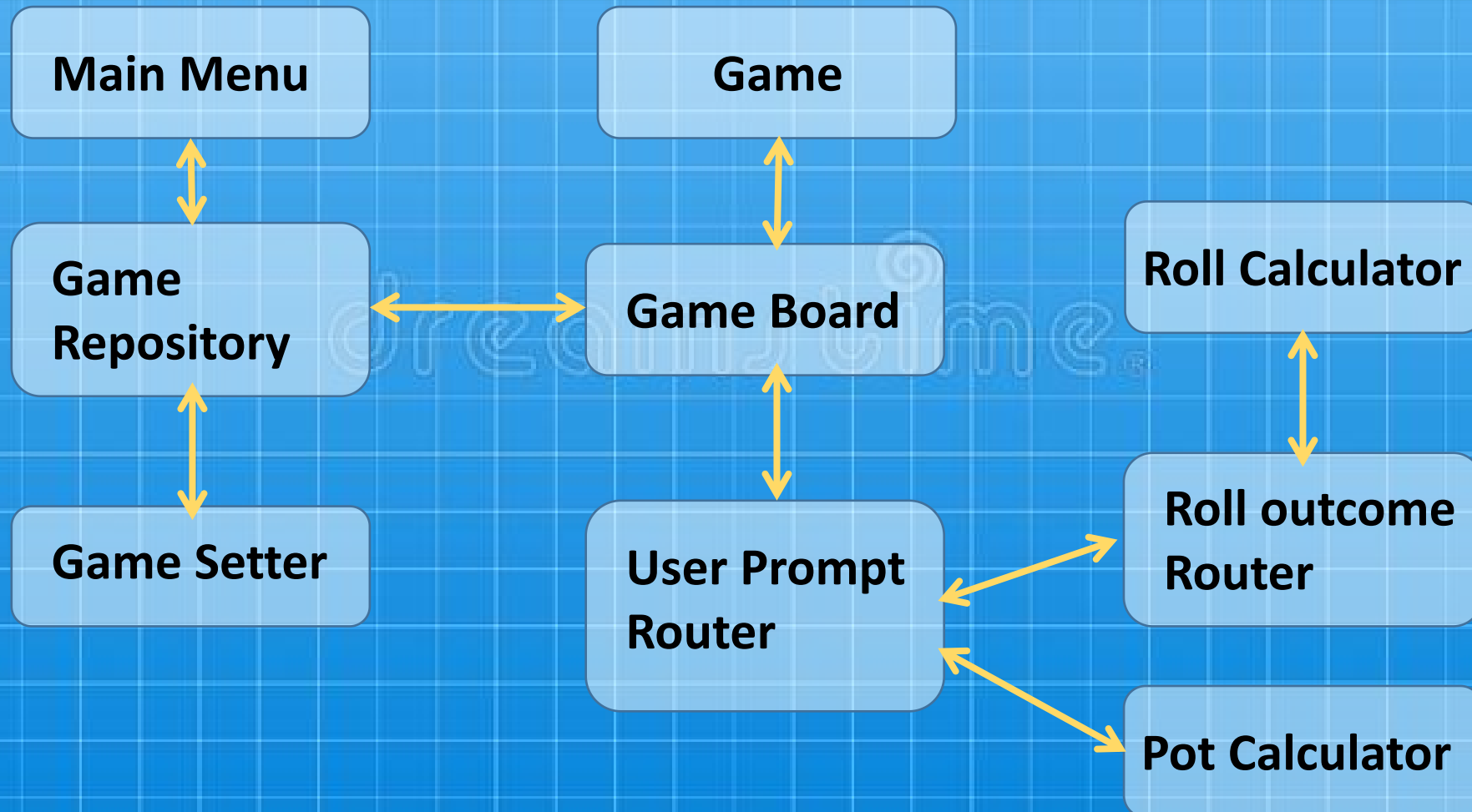
- Have attributes in only one class (DRY!)
- Check how many times you **need** to repeat instance calls
- Use inheritance properly, don't let your children run wild
- Separate conceptual concerns and expand from the general



```
src > lib > game_board.rb
1 #files
2 require_relative 'game_helper'
3
4 class GameBoard
5   include GameHelper
6
7   def initialize(game)
8     @game = game
9     @scoreboard = "Scoreboard: "
10    @instruction = "New turn, please roll all 5 dice\n\n"
11    @result = "No result to display, please select action"
12    @user_prompt = [
13      "Please select:",
14      ["Roll 5 dice", "Exit game"],
15      ['roll', 'exit'],
16      [{background: :none}, {background: :none}]
17    ]
18    @prompt_response = []
19    @current_pot = 0
20  end
21
22  def set_scoreboard
23    system("clear")
24    @scoreboard = "Scoreboard: "
25    @game.players.each do |player|
26      @scoreboard << "#{player}: #{@game.scores[player]} "
27      if player == @game.active_player
28        @scoreboard << " + #{@current_pot} "
29      end
30    end
31    @scoreboard
32  end
33
34  def set_user_prompt
35    puts ""
36  end
37 end
```

```
File Edit View Search Terminal Help
Scoreboard: Harry: 0 + [0] Tom: 0 Dick: 0
[Harry] New turn, please roll all 5 dice
No result to display, please select action
Please select: (Press ↑/↓ arrow to move and Enter to select)
> Roll 5 dice
Exit game
```


Many dependencies/redundancies



Big Problem is separation of concerns

1 Main User display

[User display code]

2 User prompt 1

3 Buiness logic 1

[User prompt code]

4 Game User display

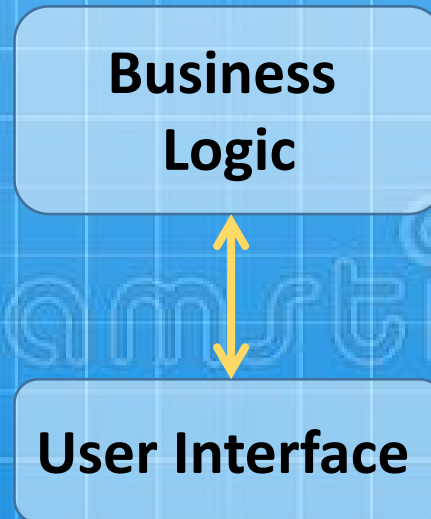
5 User prompt 2

6 Buiness logic 2

[Business logic code]

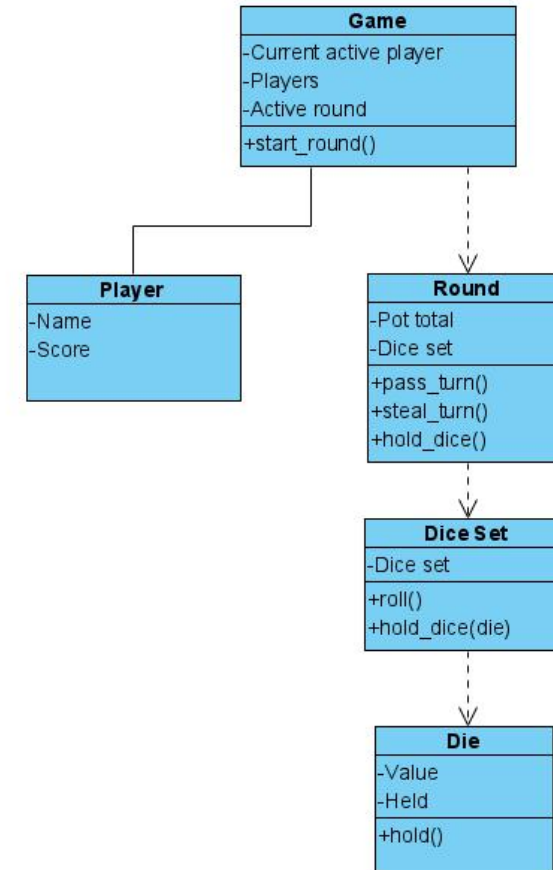
7 Game User display

8 User prompt 3...



Rudimentary Class diagram of New Game Logic

- Titles correspond to concepts/classes
- Implementation diagram that will help refine plan
- Relationship UML standard lines/arrows
- Still lots to learn and build



The background is a deep blue gradient. Overlaid on this are several white wireframe structures that resemble architectural blueprints or 3D models of buildings. These structures are composed of thin white lines forming rectangular and circular patterns. Some of the circular patterns are arranged in a grid-like fashion, while others are more scattered. The overall effect is one of a complex, futuristic, or technological environment.

Future Challenges and best parts