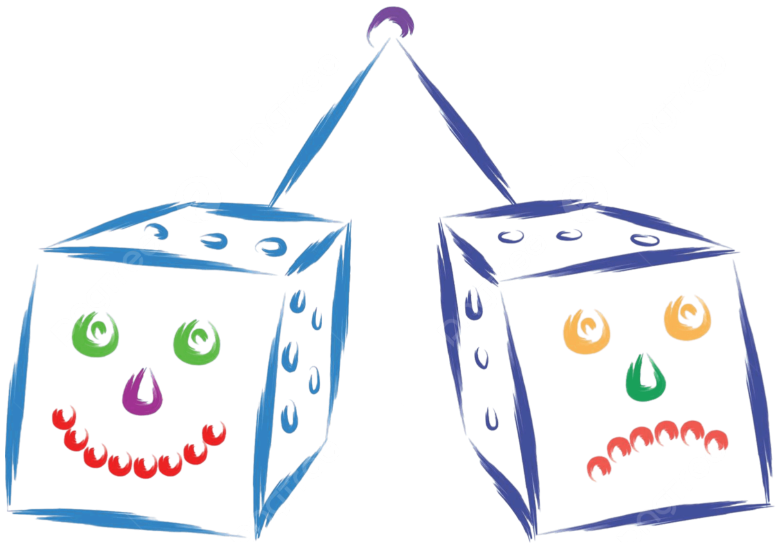


Liars Dice

Zayaan Azam

Contents

1. Premise

Five dice per player are used alongside dice cups for concealment. Players take turns rolling a “hand” of dice under their cups and make a bid based on their observation.

A bid involves a player announcing a chosen face value and the number of dice they belive show that value.

Players can either raise the bid or challenge the previous bid during their turn. Raising the bid involves increasing the quantity of dice with a specified face value.

If challenged, all dice are revealed. If the bid is valid, the bidder wins;; otherwise, the challenger wins.

The loser of a round loses one die, and the game continues until only one player retains a die or dice, declaring them the winner. The loser of the previous round initiates bidding for the next round, or the next player does so if the previous loser is eliminated.

As Per Wikipedia

2. Project Objectives

1. Program.
   1.1 `Main` - Entry point of the application. Calls `Menu.ShowMenu()` to display the main menu.
2. Constants
   2.1 `InstructionsPath` - Constant string defining the path to the instructions file.
3. Dice
   3.1 `value` - Public variable representing the current value of the dice.
   3.2 `Dice` - Constructor initializing the dice value.
   3.3 `Roll` - Method to roll the dice and return a random value.
4. Menu
   4.1 `ShowMenu` - Method displaying the main menu and handling user choices.
   4.2 `ReadInstructions` - Method to display game instructions.
5. Pirate
   5.1 `Cup` - Public list of dice representing the pirate's cup.
   5.2 `Pirate` - Constructor initializing the pirate's cup.
   5.3 `Roll` - Method to roll the dice in the cup.
   5.4 `DisplayDice` - Method to display the values of dice in the cup.
   5.5 `RaiseBid` - Abstract method for raising a bid, to be implemented by subclasses.
6. Player
   6.1 Player
   6.1 `RaiseBid` - Overrides `RaiseBid` method from `Pirate` to get bid input from the player.
7. Computer
   7.1 Computer
   7.1 `RaiseBid` - Overrides `RaiseBid` method from `Pirate` to generate a random bid for the computer player.
8. Table
   9.1 `bid` - Public variable representing the current bid.
   9.2 `player` - Public variable representing the player object.
   9.3 `computer` - Public variable representing the computer player object.
   9.4 `Table` - Constructor initializing the table.
   9.5 `GameLoop` - Method controlling the game flow.
   9.6 `CallLiar` - Method to handle when a player calls liar.
   9.7 `OutputBid` - Method to output the current bid values.

3. Documented Design

document split into 4 files.

3.1. Program.cs

intiialises the program and holds a few smaller classes.

3.2. Menu.cs

main menu screen, displays instructions.

3.3. Pirate.cs

holds the abstract class used to create the player and computer. also holds logic for each thingamajig.

3.4. Table.cs

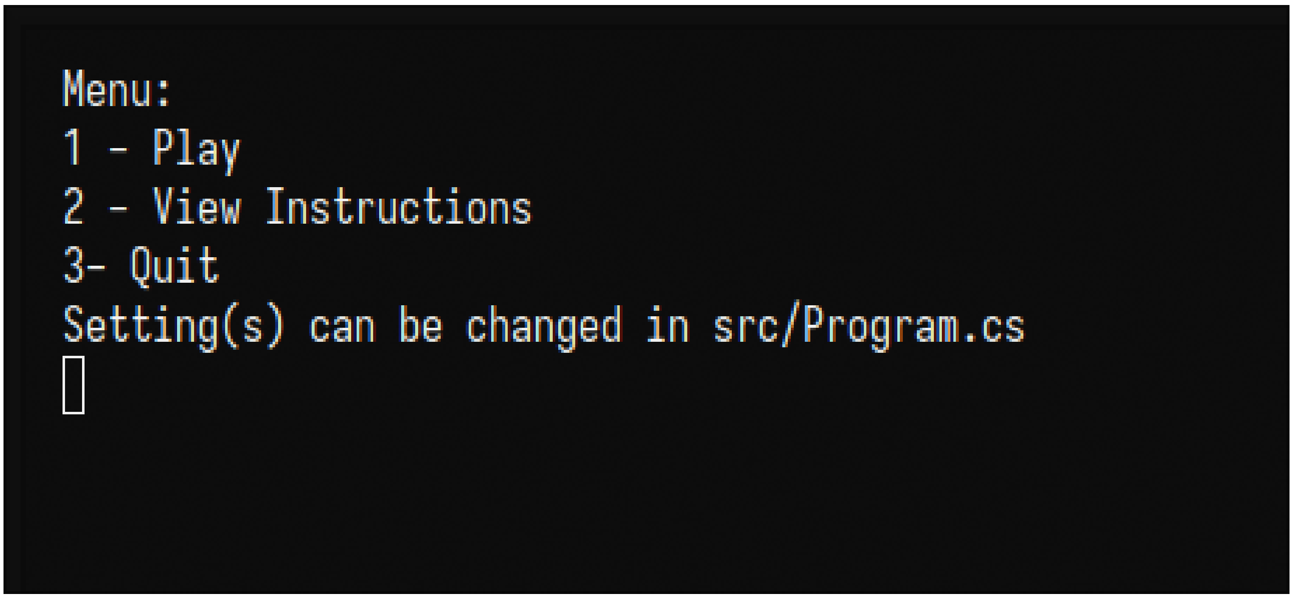
holds the game logic itself.

4. Technical Solution

seen on Github

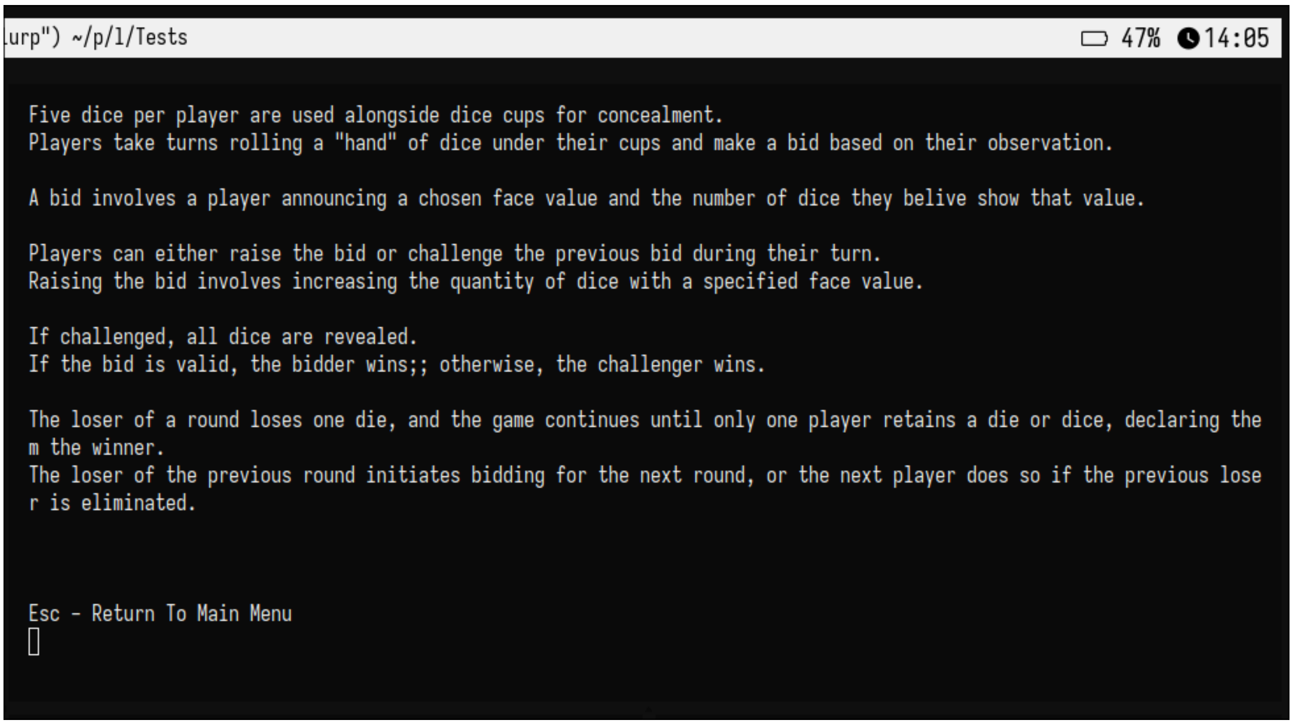
5. Testing

The program works and doesnt break it handles inputs below are some screenshots with evidence.



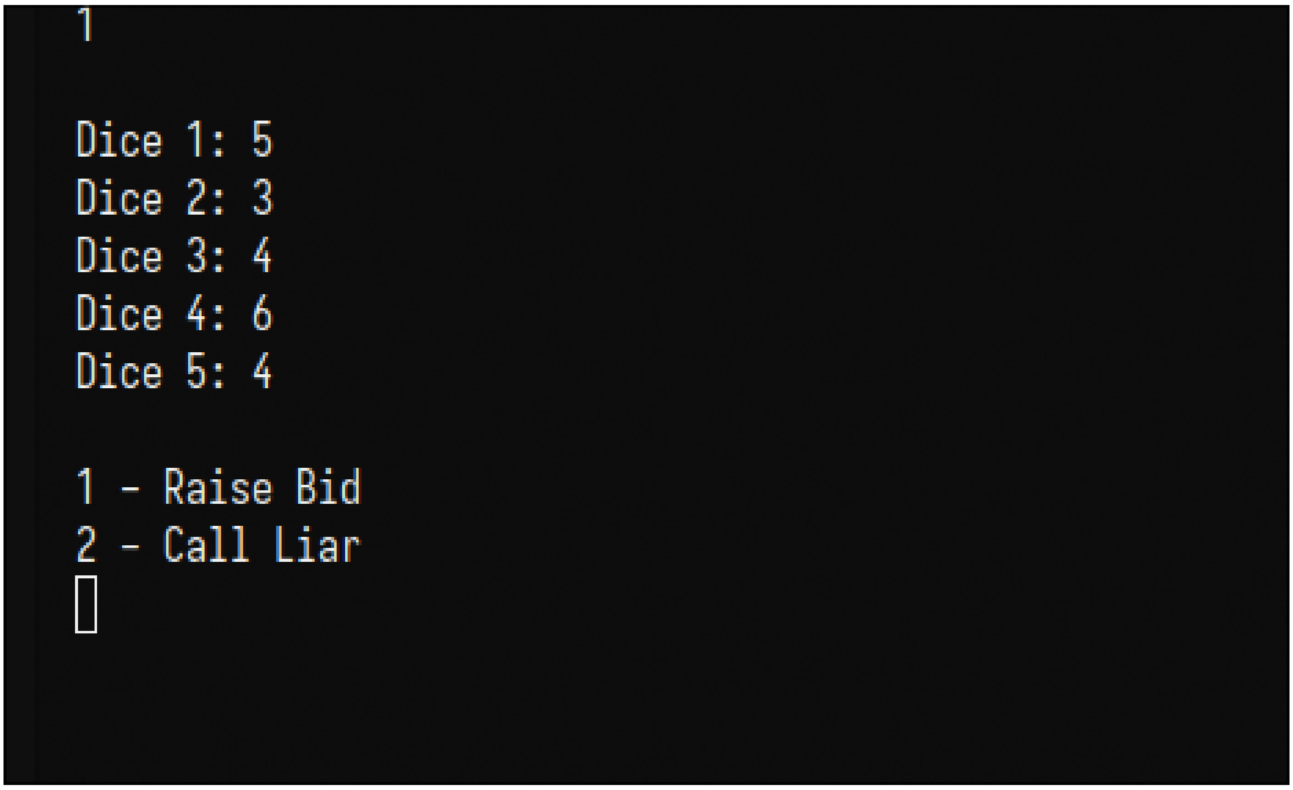
```
Menu:
1 - Play
2 - View Instructions
3- Quit
Setting(s) can be changed in src/Program.cs
```

Figure 1: Menu being outputted



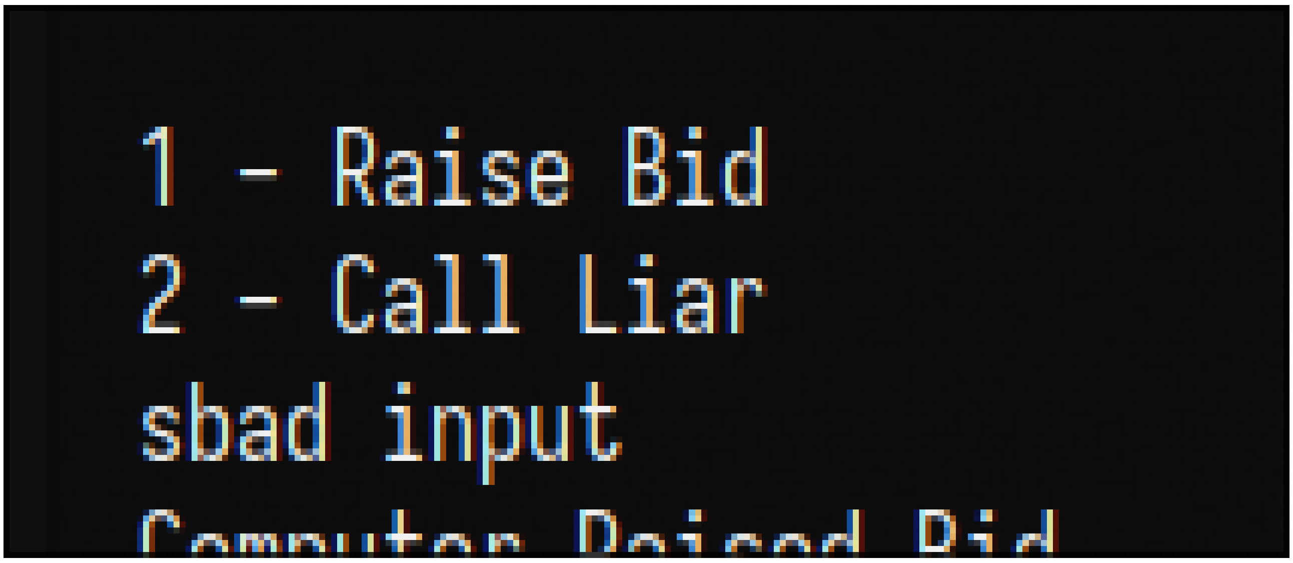
```
urp") ~/p/1/Tests                                                              47%  14:05

Five dice per player are used alongside dice cups for concealment.
Players take turns rolling a "hand" of dice under their cups and make a bid based on their observation.

A bid involves a player announcing a chosen face value and the number of dice they belive show that value.

Players can either raise the bid or challenge the previous bid during their turn.
Raising the bid involves increasing the quantity of dice with a specified face value.

If challenged, all dice are revealed.
If the bid is valid, the bidder wins;; otherwise, the challenger wins.

The loser of a round loses one die, and the game continues until only one player retains a die or dice, declaring the
m the winner.
The loser of the previous round initiates bidding for the next round, or the next player does so if the previous lose
r is eliminated.



Esc - Return To Main Menu
```

Figure 2: showcase of a submenu with the instructions being outputted



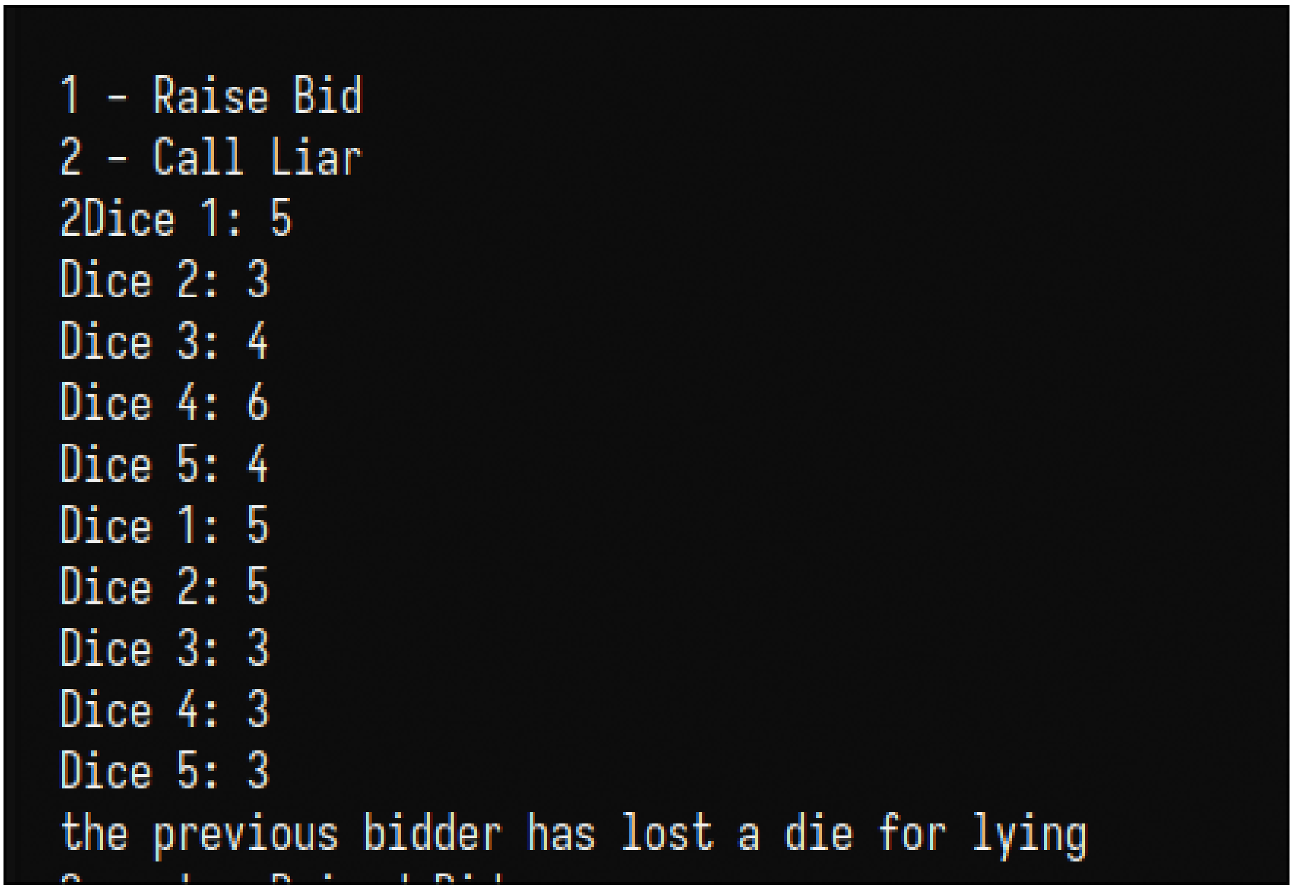
```
1

Dice 1: 5
Dice 2: 3
Dice 3: 4
Dice 4: 6
Dice 5: 4

1 - Raise Bid
2 - Call Liar
```

Figure 3: the output of a roll



```
1 - Raise Bid
2 - Call Liar
sbad input
Computer Raised Bid
```

Figure 4: error handling in case of a bad input



```
1 - Raise Bid
2 - Call Liar
2Dice 1: 5
Dice 2: 3
Dice 3: 4
Dice 4: 6
Dice 5: 4
Dice 1: 5
Dice 2: 5
Dice 3: 3
Dice 4: 3
Dice 5: 3
the previous bidder has lost a die for lying
```

Figure 5: showcase of what occurs when a liar is called

6. Evaluation

it works which is good. it addresses all project objectives

it could work better.

it handles major edge cases.

it could handle minor edge cases.

it could allow for more players / computers.

it could allow for a configuration via the menu instead of Constants.

more data could be shifted into the constants and then used

7. Bonus

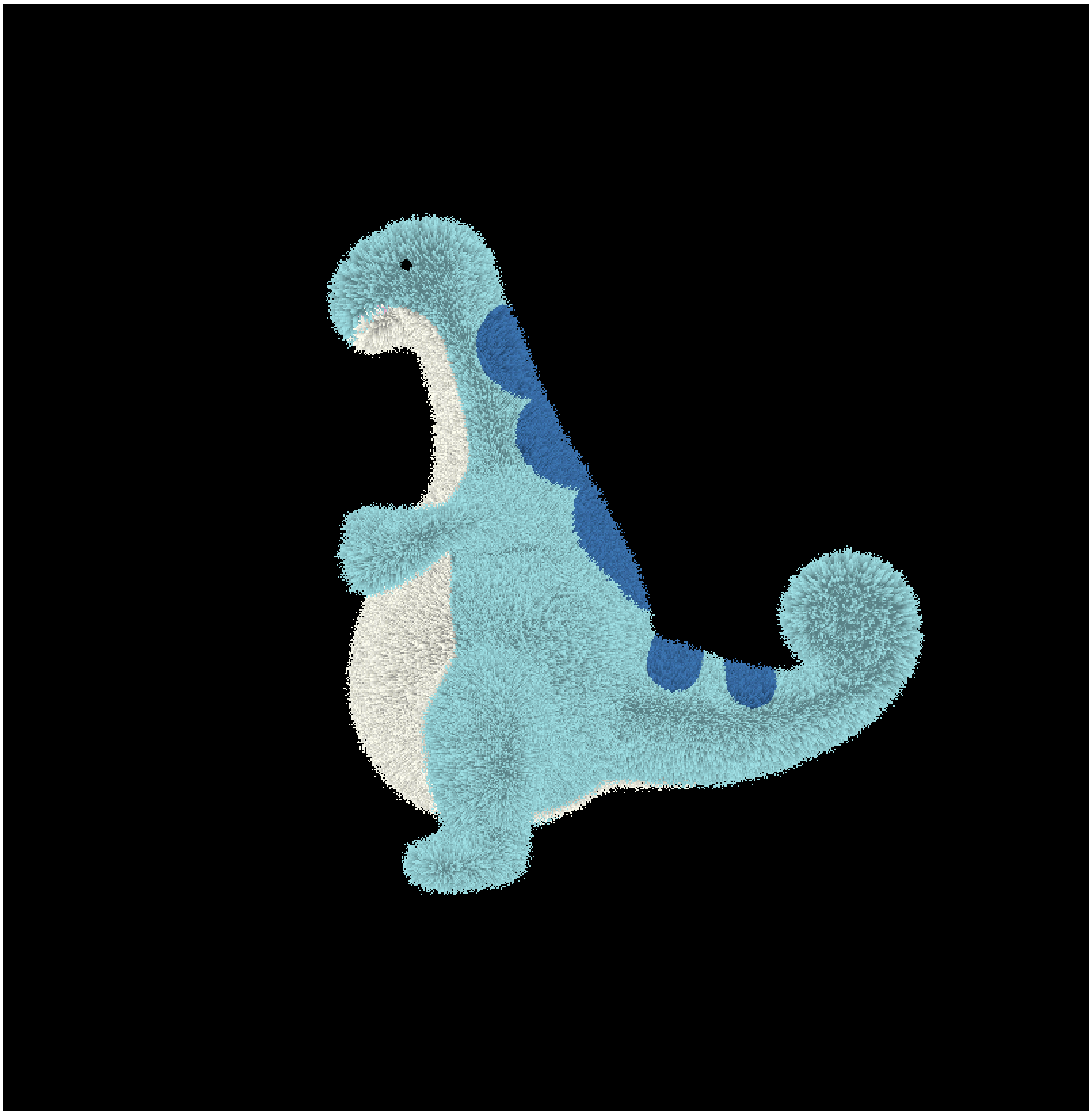


Figure 6: cool fur shader github