**\*If you have a question or comment about anything, just ask about it in the chat box or group chat**

**The Dice Game: Two-or-more-player game; take a die, add the number of points to your total, and pass it to the next player. First player to reach 20 points wins.**

**Main Game**-

**Board -** Game grid is 10x10, each grid has a rectangle. Out of the 100 rectangles, 60 of them will have a dice hidden behind them randomly. Ex, like the chart below

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \*dice |  |  | \*dice |  | \*dice | etc.. |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**Players -** Game must be played with only 2 players (might change in the future)

**Win Condition** - Instead of whoever gets exactly 20 points, it’s now whoever gets 20 points or above is the winner.

**Score Total** - Keeps tracks for each player’s points

**Changes/Mechanics to think about-**

**Timer -** Player have **X** amount of time for each move

**Pseudo Code:**

* **Step 1:** Banner, rectangle on top seperated by three pieces of info: player 1 score, player 2 score, help button

|  |  |  |
| --- | --- | --- |
| P1: Score | P2: Score | (?) |
|  |  |  |
|  |  |  |
|  |  |  |

* **Step 2:** 100 square shapes, draw a grid (done) where mouse can click on each square and get results. Note: curved edges to make it more eye-appealing  
  Possibly helpful, <https://www.youtube.com/watch?v=esKLU3dJo70>
* **Step 3:** optional - make square flip over/turn a dif color/ fade into dark + reveal item
* **Step 4:** generate 60 dice variable + 10 bonus variables at random locations - maybe assign each square in grid to array / 2 dimensional arr or (better option → ) hashmap data structure for constant time.   
  <https://processing.org/reference/HashMap.html>   
  Iterating through array to access elements will take too long. O(n) i think  
  mouse pressed at certain location, get hashmap val   
  **NOTE unfinished - still have to think abt how to create functions when dice variable, or bonus variables are hit. Then incrementing scores. Then end screen**
* End game screen: displays who won, score for each  
  Ex:   
  Player 1 Won!   
  \*gif bec why not\*   
  Player 1: 28  
  Player 2: 19

**Functionalities if we have time:**

* Start screen
* 2 player game - make it single player, or more if we have time.
* Restart button on end game screen to restart the game
* Animate dice roll

**Processing Code- (Most Recent Version of code starts from here)**

-------------------------------------------------------------------------------------------------------------------

Patrick- Added a display for remaining dice and traps. Also added a % for the remaining dice/traps.

/\*

Members Of Group 2:

BELENDEZ, STEPHANY

CHOW, PATRICK

KHAN, ALLINA

SHAPIRO, GERARD

XIAO, JASON

\*/

Rectangle[][] rectangle = new Rectangle[6][10];

// if one = true, its player 1's turn, else its player 2's turn

boolean one, gameEnd = false, restart = false;

int oneScore, twoScore, game = 0, gameStartTime = 0, numDice = 0, numTraps = 0;

class Rectangle

{

int xAxis, yAxis, dice;

//Constructor to set the rectangle's X and Y axis

Rectangle(int x, int y)

{

xAxis = x;

yAxis = y;

}

void diceRoll(int d)

{

dice = d;

}

//draw the rectangle object

void drawRect()

{

rect(xAxis, yAxis, 80, 70, 12);

}

// turn this rectangle to black

// change the dice value to -1 to avoid being clicked twice

// return the dice value

int clickRect()

{

// if this rectangle had been clicked before, return

if(dice == -1)

return dice;

// if dice = 1, then there is a dice, generate a random number between 1-6

int roll = dice;

if(dice == 1)

roll = int(random(6)+1);

dice = -1;

fill(0);

rect(xAxis, yAxis, 80, 70, 12);

//displays score clicked

showScore(xAxis,yAxis,roll);

return roll;

}

}

void setup()

{

size(800,520);

background(0);

stroke(255);

strokeWeight(3);

textSize(30);

text("How To Play:\nEach player take turns selecting a rectangle." +

"\nThere are 3 kinds of rectangle: Dice, Trap, and Blank." +

"\n\nDice gives random points from 1-6 when clicked." +

"\nTrap gives negative value between -4 to -2." +

"\nBlank rectangle gives nothing." +

"\nThe first player to each a score above 20 wins." +

"\nPress 'R' to begin the game.", CENTER,50);

//Used to fill the field up Rectangle objects

for(int xLocation = 0; xLocation < 10; xLocation++) //loop for columns

{

for(int yLocation = 0; yLocation < 6; yLocation++) //loop for rows

rectangle[yLocation][xLocation] = new Rectangle(xLocation\*80, yLocation\*70+100);

}

}

void draw()

{

if(restart)

{

int temp =0;

temp +=millis();

game = 1;

oneScore = twoScore = numDice = numTraps = 0;

one = true;

gameEnd = false;

restart = false;

background(0);

stroke(255);

line(0, 95, width, 95); //Create a line to seperate the banner and the field

//Separate the players' score from the information box

//and adds an outline for the program

line(200,0, 200, 95);

line(0,0,0,95);

line(0,0,width,0);

line(width,0, width, 95);

//Display the players' score (Formatting must be done like this for exact precision)

fill(255, 0, 0);

text("Player 1: ", width\*.25-185, height-485);

fill(255);

text(oneScore, width\*.25-55, height-485);

text("Player 2:", width\*.25-185, height-447);

text(twoScore, width\*.25-55, height-447);

//Draw the rectangles

stroke(0);

for(int x = 0; x < 10; x++)

{

for(int y = 0; y < 6; y++)

{

rectangle[y][x].drawRect();

rectangle[y][x].diceRoll(0);

}

}

// generate the dice and the traps

for(int a = 0; a < 50; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(1);

for(int a = 0; a < 10; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(int(random(-4, -2)));

//Count the number of dice and traps remaining

countDiceAndTraps(rectangle);

//Display the number of dice and traps remaining

displayDiceAndTraps();

}

// if the game ends, display the end game screen

if(gameEnd)

{

background(0);

fill(255);

if(oneScore > twoScore)

text("Player 1 Won!", 290, 100);

else

text("Player 2 Won!", 290, 100);

text("Player 1: " + oneScore + "\nPlayer 2: " + twoScore, 300, 150);

text("Press 'R' to start a new game", 190, 250);

}

// if any player's score reached 20, end the game

if(oneScore >= 20 || twoScore >= 20)

gameEnd = true;

}

void mousePressed()

{

if(gameEnd || game == 0)

return;

if(mouseY > 100)

{

// calculate the index of the rectangle based on where the player clicks

// without the int(), it won’t work on openprocessing

int x = int(mouseX/80);

int y = int((mouseY-100)/70);

int roll = rectangle[y][x].clickRect();

// if roll == -1, it means that this rectangle had already been clicked

if(roll == -1)

return;

// update the score of the player

// the rect() is for erasing the previous score

fill(0);

if(one)

{

int previousScore = oneScore;

oneScore += roll;

//Used to keep track of the number of Traps

if(previousScore > oneScore)

numTraps--;

//Used to keep track of the number of Dice

if(oneScore > previousScore)

numDice--;

//Prevent player 1 from having a negative score

if(oneScore <=0)

oneScore = 0;

one = false;

fill(0);

rect(5, 5, 180 , 80);

//Color code player two (since player 1 is initially color-coded)

fill(255);

text("Player 1: ", width\*.25-185, height-485);

text(oneScore, width-655, height-485);

fill(0,0,255);

text("Player 2:", width\*.25-185, height-447);

fill(255);

text(twoScore, width\*.25-55, height-447);

}

else

{

int previousScore = twoScore;

twoScore += roll;

//Used to keep track of the number of Traps

if(previousScore > twoScore)

numTraps--;

//Used to keep track of the number of Dice

if(twoScore > previousScore)

numDice--;

//Prevent player 2 from having a negative score

if(twoScore <=0)

twoScore = 0;

one = true;

rect(5, 5, 180 , 80);

//Color code player one

fill(255,0,0);

text("Player 1: ", width\*.25-185, height-485);

fill(255);

text(oneScore, width-655, height-485);

fill(255);

text("Player 2:", width\*.25-185, height-447);

text(twoScore, width\*.25-55, height-447);

}

//Count the number of dice and traps remaining and display them

displayDiceAndTraps();

}

}

void keyPressed()

{

if(key == 'r' || key == 'R')

restart = true;

}

void showScore(int x, int y, int roll)

{

//color codes score dependent on who revealed it

//red=player1 blue=player2

if(one == true)

fill(255,0,0);

else

fill(0,0,255);

text(roll,x+30,y+50);

}

//Count the number of dice and traps remaining

void countDiceAndTraps(Rectangle rectangle[][])

{

//Count the number of dice and traps remaining

for(int yIndex = 0; yIndex<6; yIndex++)

{

for(int xIndex = 0; xIndex <10; xIndex++)

{

if(rectangle[yIndex][xIndex].dice>0)

{

numDice++;

}

if(rectangle[yIndex][xIndex].dice<-1)

{

numTraps++;

}

}

}

}

//Display the number of dice and traps remaining

void displayDiceAndTraps()

{

fill(0);

rect(203,3,350,85);

fill(255);

text("Remaining Dice: " + numDice, 210,37);

text("Remaining Traps: " + numTraps, 210,73);

}

---------------------------------------------------------------------------------------------------------------------------------------------------

Stephany: I added showScore() to show the number under the clicked rectangle. It is color-coded dependent on who revealed it (p1=red || p2=blue). Score remains on screen throughout gameplay. Also, adjusted the color-coding in MousePressed(), since player 1 always begins and is therefore initially color-coded, when the mouse is pressed for the first time that would conclude player 1's turn, and so we must colorcode player two to indicate that it’s their turn.

Rectangle[][] rectangle = new Rectangle[6][10];

// if one = true, its player 1's turn, else its player 2's turn

boolean one, gameEnd = false, restart = false;

int oneScore, twoScore, game = 0, gameStartTime = 0;

class Rectangle

{

int xAxis, yAxis, dice;

//Constructor to set the rectangle's X and Y axis

Rectangle(int x, int y)

{

xAxis = x;

yAxis = y;

}

void diceRoll(int d)

{

dice = d;

}

//draw the rectangle object

void drawRect()

{

rect(xAxis, yAxis, 80, 70, 12);

}

// turn this rectangle to black

// change the dice value to -1 to avoid being clicked twice

// return the dice value

int clickRect()

{

// if this rectangle had been clicked before, return

if(dice == -1)

return dice;

// if dice = 1, then there is a dice, generate a random number between 1-6

int roll = dice;

if(dice == 1)

roll = int(random(6)+1);

dice = -1;

fill(0);

rect(xAxis, yAxis, 80, 70, 12);

//displays score clicked

showScore(xAxis,yAxis,roll);

return roll;

}

}

void setup()

{

size(800,520);

background(0);

stroke(255);

strokeWeight(3);

textSize(30);

text("How To Play:\nEach player take turns selecting a rectangle." +

"\nThere are 3 kinds of rectangle: Dice, Trap, and Blank." +

"\n\nDice gives random points from 1-6 when clicked." +

"\nTrap gives negative value between -4 to -2." +

"\nBlank rectangle gives nothing." +

"\nThe first player to each a score above 20 wins." +

"\nPress 'R' to begin the game.", CENTER,50);

//Used to fill the field up Rectangle objects

for(int xLocation = 0; xLocation < 10; xLocation++) //loop for columns

{

for(int yLocation = 0; yLocation < 6; yLocation++) //loop for rows

rectangle[yLocation][xLocation] = new Rectangle(xLocation\*80, yLocation\*70+100);

}

}

void draw()

{

if(restart)

{

int temp =0;

temp +=millis();

game = 1;

oneScore = twoScore = 0;

one = true;

gameEnd = false;

restart = false;

background(0);

stroke(255);

line(0, 95, width, 95); //Create a line to seperate the banner and the field

//Separate the players' score from the information box

//and adds an outline for the program

line(200,0, 200, 95);

line(0,0,0,95);

line(0,0,width,0);

line(width,0, width, 95);

//Display the players' score (Formatting must be done like this for exact precision)

fill(255, 0, 0);

text("Player 1: ", width\*.25-185, height-485);

fill(255);

text(oneScore, width\*.25-55, height-485);

text("Player 2:", width\*.25-185, height-447);

text(twoScore, width\*.25-55, height-447);

stroke(0);

for(int x = 0; x < 10; x++)

{

for(int y = 0; y < 6; y++)

{

rectangle[y][x].drawRect();

rectangle[y][x].diceRoll(0);

}

}

// generate the dice and the traps

for(int a = 0; a < 40; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(1);

for(int a = 0; a < 10; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(int(random(-4, -2)));

}

// if the game ends, display the end game screen

if(gameEnd)

{

background(0);

fill(255);

if(oneScore > twoScore)

text("Player 1 Won!", 290, 100);

else

text("Player 2 Won!", 290, 100);

text("Player 1: " + oneScore + "\nPlayer 2: " + twoScore, 300, 150);

text("Press 'R' to start a new game", 190, 250);

}

// if any player's score reached 20, end the game

if(oneScore >= 20 || twoScore >= 20)

gameEnd = true;

}

void mousePressed()

{

if(gameEnd || game == 0)

return;

if(mouseY > 100)

{

// calculate the index of the rectangle based on where the player clicks

// without the int(), it won’t work on openprocessing

int x = int(mouseX/80);

int y = int((mouseY-100)/70);

int roll = rectangle[y][x].clickRect();

// if roll == -1, it means that this rectangle had already been clicked

if(roll == -1)

return;

// update the score of the player

// the rect() is for erasing the previous score

fill(0);

if(one)

{

oneScore += roll;

//Prevent player 1 from having a negative score

if(oneScore <=0)

oneScore = 0;

one = false;

fill(0);

rect(5, 5, 180 , 80);

//Color code player two (since player 1 is initially color-coded)

fill(255);

text("Player 1: ", width\*.25-185, height-485);

text(oneScore, width-655, height-485);

fill(0,0,255);

text("Player 2:", width\*.25-185, height-447);

fill(255);

text(twoScore, width\*.25-55, height-447);

}

else

{

twoScore += roll;

//Prevent player 2 from having a negative score

if(twoScore <=0)

twoScore = 0;

one = true;

rect(5, 5, 180 , 80);

//Color code player one

fill(255,0,0);

text("Player 1: ", width\*.25-185, height-485);

fill(255);

text(oneScore, width-655, height-485);

fill(255);

text("Player 2:", width\*.25-185, height-447);

text(twoScore, width\*.25-55, height-447);

}

}

}

void keyPressed()

{

if(key == 'r' || key == 'R')

restart = true;

}

void showScore(int x, int y, int roll)

{

//color codes score dependent on who revealed it

//red=player1 blue=player2

if(one == true)

fill(255,0,0);

else

fill(0,0,255);

text(roll,x+30,y+50);

}

**------------------------------------------------------------------------------------------------------------**

**Patrick - I updated the code to properly adjust to the change in height, fixed the bug where sometimes clicking a rectangle wouldn’t work properly, made it so the traps can only have the value between -4 to -2, made it so that no player can have a negative score, moved the player statuses to the top left to add room for a description box. (Not Yet Implemented) The description box will say things such as “Player 1 rolled a 3” or “Player 1 fell into a trap and lost 2 points”. It can also have things such as the total amount of turns that have passed and the number of rectangles with dice/trap behind them currently.**

**Rectangle[][] rectangle = new Rectangle[6][10];**

**// if one = true, its player 1's turn, else its player 2's turn**

**boolean one, gameEnd = false, restart = false;**

**int oneScore, twoScore, game = 0, gameStartTime = 0;**

**class Rectangle**

**{**

**int xAxis, yAxis, dice;**

**//Constructor to set the rectangle's X and Y axis**

**Rectangle(int x, int y)**

**{**

**xAxis = x;**

**yAxis = y;**

**}**

**void diceRoll(int d)**

**{**

**dice = d;**

**}**

**//draw the rectangle object**

**void drawRect()**

**{**

**rect(xAxis, yAxis, 80, 70, 12);**

**}**

**// turn this rectangle to black**

**// change the dice value to -1 to avoid being clicked twice**

**// return the dice value**

**int clickRect()**

**{**

**// if this rectangle had been clicked before, return**

**if(dice == -1)**

**return dice;**

**// if dice = 1, then there is a dice, generate a random number between 1-6**

**int roll = dice;**

**if(dice == 1)**

**roll = int(random(6)+1);**

**dice = -1;**

**fill(0);**

**rect(xAxis, yAxis, 80, 70, 12);**

**return roll;**

**}**

**}**

**void setup()**

**{**

**size(800,520);**

**background(0);**

**stroke(255);**

**strokeWeight(3);**

**textSize(30);**

**text("How To Play:\nEach player take turns selecting a rectangle." +**

**"\nThere are 3 kinds of rectangle: Dice, Trap, and Blank." +**

**"\n\nDice gives random points from 1-6 when clicked." +**

**"\nTrap gives negative value between -4 to -2." +**

**"\nBlank rectangle gives nothing." +**

**"\nThe first player to each a score above 20 wins." +**

**"\nPress 'R' to begin the game.", CENTER,50);**

**//Used to fill the field up Rectangle objects**

**for(int xLocation = 0; xLocation < 10; xLocation++) //loop for columns**

**{**

**for(int yLocation = 0; yLocation < 6; yLocation++) //loop for rows**

**rectangle[yLocation][xLocation] = new Rectangle(xLocation\*80, yLocation\*70+100);**

**}**

**}**

**void draw()**

**{**

**if(restart)**

**{**

**int temp =0;**

**temp +=millis();**

**game = 1;**

**oneScore = twoScore = 0;**

**one = true;**

**gameEnd = false;**

**restart = false;**

**background(0);**

**stroke(255);**

**line(0, 95, width, 95); //Create a line to seperate the banner and the field**

**//Separate the players' score from the information box**

**//and adds an outline for the program**

**line(200,0, 200, 95);**

**line(0,0,0,95);**

**line(0,0,width,0);**

**line(width,0, width, 95);**

**//Display the players' score (Formatting must be done like this for exact precision)**

**fill(255, 0, 0);**

**text("Player 1: ", width\*.25-185, height-485);**

**fill(255);**

**text(oneScore, width\*.25-55, height-485);**

**text("Player 2:", width\*.25-185, height-447);**

**text(twoScore, width\*.25-55, height-447);**

**stroke(0);**

**for(int x = 0; x < 10; x++)**

**{**

**for(int y = 0; y < 6; y++)**

**{**

**rectangle[y][x].drawRect();**

**rectangle[y][x].diceRoll(0);**

**}**

**}**

**// generate the dice and the traps**

**for(int a = 0; a < 40; a++)**

**rectangle[int(random(6))][int(random(10))].diceRoll(1);**

**for(int a = 0; a < 10; a++)**

**rectangle[int(random(6))][int(random(10))].diceRoll(int(random(-4, -2)));**

**}**

**// if the game ends, display the end game screen**

**if(gameEnd)**

**{**

**background(0);**

**fill(255);**

**if(oneScore > twoScore)**

**text("Player 1 Won!", 290, 100);**

**else**

**text("Player 2 Won!", 290, 100);**

**text("Player 1: " + oneScore + "\nPlayer 2: " + twoScore, 300, 150);**

**text("Press 'R' to start a new game", 190, 250);**

**}**

**}**

**void mousePressed()**

**{**

**// if any player's score reached 20, end the game**

**if(oneScore >= 20 || twoScore >= 20)**

**gameEnd = true;**

**if(gameEnd || game == 0)**

**return;**

**if(mouseY > 100)**

**{**

**// calculate the index of the rectangle based on where the player clicks**

**// without the int(), it won’t work on openprocessing**

**int x = int(mouseX/80);**

**int y = int((mouseY-100)/70);**

**int roll = rectangle[y][x].clickRect();**

**// if roll == -1, it means that this rectangle had already been clicked**

**if(roll == -1)**

**return;**

**// update the score of the player**

**// the rect() is for erasing the previous score**

**fill(0);**

**if(one)**

**{**

**oneScore += roll;**

**//Prevent player 1 from having a negative score**

**if(oneScore <=0)**

**oneScore = 0;**

**one = false;**

**fill(0);**

**rect(5, 5, 180 , 80);**

**//Color code player one**

**fill(255,0,0);**

**text("Player 1: ", width\*.25-185, height-485);**

**fill(255);**

**text(oneScore, width-655, height-485);**

**text("Player 2:", width\*.25-185, height-447);**

**text(twoScore, width\*.25-55, height-447);**

**}**

**else**

**{**

**twoScore += roll;**

**//Prevent player 2 from having a negative score**

**if(twoScore <=0)**

**twoScore = 0;**

**one = true;**

**rect(5, 5, 180 , 80);**

**//Color code player two**

**fill(255);**

**text("Player 1: ", width\*.25-185, height-485);**

**text(oneScore, width-655, height-485);**

**fill(0,0,255);**

**text("Player 2:", width\*.25-185, height-447);**

**fill(255);**

**text(twoScore, width\*.25-55, height-447);**

**}**

**}**

**}**

**void keyPressed()**

**{**

**if(key == 'r' || key == 'R')**

**restart = true;**

**}**

**-----------------------------------------------------------------------------------------**

//teamwork patrick allina stephany - changed the box size 6\*10 grid

--------------------------------------------------------------------------------------------------------------------------------

Rectangle[][] rectangle = new Rectangle[6][10];

// if one = true, its player 1's turn, else its player 2's turn

boolean one, gameEnd = false, restart = false;

int oneScore, twoScore, game = 0;

class Rectangle

{

int xAxis, yAxis, dice;

//Constructor to set the rectangle's X and Y axis

Rectangle(int x, int y)

{

xAxis = x;

yAxis = y;

}

void diceRoll(int d)

{

dice = d;

}

//draw the rectangle object

void drawRect()

{

rect(xAxis, yAxis, 80, 70, 12);

}

// turn this rectangle to black

// change the dice value to -1 to avoid being clicked twice

// return the dice value

int clickRect()

{

// if this rectangle had been clicked before, return

if(dice == -1)

return dice;

// if dice = 1, then there is a dice, generate a random number between 1-6

int roll = dice;

if(dice == 1)

roll = int(random(6)+1);

dice = -1;

fill(0);

rect(xAxis, yAxis, 80, 70, 12);

return roll;

}

}

void setup()

{

size(800,520);

background(0);

stroke(255);

strokeWeight(3);

textSize(30);

text("How To Play:\nEach player take turns selecting a rectangle." +

"\nThere are 3 kinds of rectangle: Dice, Trap, and Blank." +

"\n\nDice gives random points from 1-6 when clicked." +

"\nTrap gives negative value between -10 to -2." +

"\nBlank rectangle gives nothing." +

"\nThe first player to each a score above 20 wins." +

"\nPress 'R' to begin the game.", CENTER,150);

//Used to fill the field up Rectangle objects

for(int xLocation = 0; xLocation < 10; xLocation++) //loop for columns

{

for(int yLocation = 0; yLocation < 6; yLocation++) //loop for rows

rectangle[yLocation][xLocation] = new Rectangle(xLocation\*80, yLocation\*70+100);

}

}

void draw()

{

if(restart)

{

game = 1;

oneScore = twoScore = 0;

one = true;

gameEnd = false;

restart = false;

background(0);

stroke(255);

line(0, 95, width, 95); //Create a line to seperate the banner and the field

line(0, 95, width, 95); //Create a line to seperate the banner and the field

line(190,0, 190, 95); //Separate the players' score from the information box

//Display the players' score

fill(255, 0, 0);

text("Player 1: ", width\*.25-185, height-485);

fill(255);

text("Player 2: ", width\*.25-185, height-447);

stroke(0);

for(int x = 0; x < 10; x++)

{

for(int y = 0; y < 6; y++)

{

rectangle[y][x].drawRect();

rectangle[y][x].diceRoll(0);

}

}

// generating dice and the trap

for(int a = 0; a < 60; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(1);

for(int a = 0; a < 10; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(int(random(-10, -1)));

}

// if the game ends, display the end game screen

if(gameEnd)

{

background(0);

fill(255);

if(oneScore > twoScore)

text("Player 1 Won!", 290, 300);

else

text("Player 2 Won!", 290, 300);

text("Player 1: " + oneScore + "\nPlayer 2: " + twoScore, 300, 350);

text("Press 'R' to start a new game", 190, 450);

}

}

void mousePressed()

{

if(gameEnd || game == 0)

return;

if(mouseY > 100)

{

// calculate the index of the rectangle based on where the player clicks

// without the int(), it won’t work on openprocessing

int x = int(mouseX/80);

int y = int((mouseY-100)/70);

int roll = rectangle[y][x].clickRect();

// if roll == -1, it means that this rectangle had already been clicked

if(roll == -1)

return;

// update the score of the player

// the rect() is for erasing the previous score

fill(0);

if(one)

{

oneScore += roll;

one = false;

rect(225, 30, 100, 50);

fill(255);

text(oneScore, 240, 60);

text("Player 1: ", width\*.25-100, height-740);

fill(0, 0, 255);

text("Player 2: ", width\*.75-100, height-740);

}

else

{

twoScore += roll;

one = true;

rect(625, 30, 100, 50);

fill(255);

text(twoScore, 640, 60);

text("Player 2: ", width\*.75-100, height-740);

fill(255, 0, 0);

text("Player 1: ", width\*.25-100, height-740);

}

}

// if any player's score reached 20, end the game

if(oneScore >= 20 || twoScore >= 20)

gameEnd = true;

}

void keyPressed()

{

if(key == 'r' || key == 'R')

restart = true;

}

---------------------------------------------------------------------------------------------------------------------------------------------------

#### // Jason - I implemented the description box that Patrick mentioned, and added some functions so the rectangles would show the points and which player flipped it.

Rectangle[][] rectangle = new Rectangle[6][10];

// if one = true, its player 1's turn, else its player 2's turn

boolean one, gameEnd = false, restart = false;

int oneScore, twoScore, turn, game = 0, gameStartTime = 0;

class Rectangle

{

int xAxis, yAxis, dice;

//Constructor to set the rectangle's X and Y axis

Rectangle(int x, int y)

{

xAxis = x;

yAxis = y;

}

void diceRoll(int d)

{

dice = d;

}

//draw the rectangle object

void drawRect()

{

rect(xAxis, yAxis, 80, 70, 12);

}

// turn this rectangle to black

// change the dice value to -1 to avoid being clicked twice

// return the dice value

int clickRect()

{

// if this rectangle had been clicked before, return

if(dice == -1)

return dice;

// if dice = 1, then there is a dice, generate a random number between 1-6

int roll = dice;

if(dice == 1)

roll = int(random(6)+1);

dice = -1;

if(one)

fill(255, 0, 0);

else

fill(0, 0, 255);

rect(xAxis, yAxis, 80, 70, 12);

fill(255);

text(roll, xAxis+30, yAxis+40);

return roll;

}

}

void setup()

{

size(800,520);

background(0);

stroke(255);

strokeWeight(3);

textSize(30);

text("How To Play:\nEach player take turns selecting a rectangle." +

"\nThere are 3 kinds of rectangle: Dice, Trap, and Blank." +

"\n\nDice gives random points from 1-6 when clicked." +

"\nTrap gives negative value between -4 to -2." +

"\nBlank rectangle gives nothing." +

"\nThe first player to each a score above 20 wins." +

"\nPress 'R' to begin the game.", CENTER, 100);

//Used to fill the field up Rectangle objects

for(int xLocation = 0; xLocation < 10; xLocation++) //loop for columns

{

for(int yLocation = 0; yLocation < 6; yLocation++) //loop for rows

rectangle[yLocation][xLocation] = new Rectangle(xLocation\*80, yLocation\*70+100);

}

}

void draw()

{

if(restart)

{

turn = game = 1;

oneScore = twoScore = 0;

one = true;

gameEnd = restart = false;

background(0);

stroke(255);

line(0, 95, width, 95); //Create a line to seperate the banner and the field

//Separate the players' score from the information box

//and adds an outline for the program

line(200, 0, 200, 95);

line(0, 0, 0, 95);

line(0, 0, width, 0);

line(width, 0, width, 95);

//Display the players' score (Formatting must be done like this for exact precision)

fill(255, 0, 0);

text("Player 1: ", width\*.25-185, height-485);

fill(255);

text(oneScore, width\*.25-55, height-485);

text("Player 2:", width\*.25-185, height-447);

text(twoScore, width\*.25-55, height-447);

text("Turn " + turn++ + ": Game Start", 220, 55);

stroke(0);

for(int x = 0; x < 10; x++)

{

for(int y = 0; y < 6; y++)

{

rectangle[y][x].drawRect();

rectangle[y][x].diceRoll(0);

}

}

// generate the dice and the traps

for(int a = 0; a < 40; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(1);

for(int a = 0; a < 10; a++)

rectangle[int(random(6))][int(random(10))].diceRoll(int(random(-4, -1)));

}

// if the game ends, display the end game screen

if(gameEnd)

{

background(0);

fill(255);

if(oneScore > twoScore)

text("Player 1 Won!", 290, 150);

else

text("Player 2 Won!", 290, 150);

text("Player 1: " + oneScore + "\nPlayer 2: " + twoScore, 300, 200);

text("Press 'R' to start a new game", 190, 300);

}

}

void mousePressed()

{

// if any player's score reached 20, end the game

if(oneScore >= 20 || twoScore >= 20)

gameEnd = true;

if(gameEnd || game == 0)

return;

if(mouseY > 100)

{

// calculate the index of the rectangle based on where the player clicks

// without int(), it won’t work on openprocessing

int x = int(mouseX/80);

int y = int((mouseY-100)/70);

int roll = rectangle[y][x].clickRect();

// if roll == -1, it means that this rectangle had already been clicked

if(roll == -1)

return;

// update the game state to the screen

// the rect() is for covering the previous text

fill(0);

rect(5, 5, 180, 80);

rect(210, 5, 585, 80);

if(one)

{

oneScore += roll;

//Prevent player 1 from having a negative score

if(oneScore <= 0)

oneScore = 0;

one = false;

fill(255,0,0);

text("Player 1: ", width\*.25-185, height-485);

fill(255);

text("Player 2:", width\*.25-185, height-447);

}

else

{

twoScore += roll;

//Prevent player 2 from having a negative score

if(twoScore <= 0)

twoScore = 0;

one = true;

fill(0,0,255);

text("Player 2:", width\*.25-185, height-447);

fill(255);

text("Player 1: ", width\*.25-185, height-485);

}

text(oneScore, width-655, height-485);

text(twoScore, width\*.25-55, height-447);

text(description(roll), 220, 55);

}

}

String description(int roll)

{

String description = "";

if(one)

description = description + "Turn " + turn++ + ": Player 2 ";

else

description = description + "Turn " + turn++ + ": Player 1 ";

if(roll > 0)

description = description + "gained " + roll + " points";

else if(roll < 0)

description = description + "fell into a trap, minus " + abs(roll);

else

description = description + "clicked a blank rectangle";

return description;

}

void keyPressed()

{

if(key == 'r' || key == 'R')

restart = true;

}

-------------------------------------------------------------------------------------------------------------

//Patrick- Took Jason’s code and formatted it better. Added instructions to the start screen and made it color //coded to help know whose turn it currently is.

//2 bugs currently exist right now.

//Most important is that the code doesn’t work on open processing but works fine on processing.

//Other one is that some of the instructions text stay on the screen after the game starts (Line 65)

Rectangle[][] rectangle = new Rectangle[10][10];

// if one = true, its player 1's turn, else its player 2's turn

boolean one = true, gameEnd = false;

int oneScore = 0, twoScore = 0, game = 0;

void setup()

{

size(800,800);

background(0);

stroke(255);

strokeWeight(3);

line(0,95,width,95); //Create a line to seperate the banner and the field

textSize(30);

}

void draw()

{

//display the instructions on how to play the game

instructions();

//Used as a counter to initalize the board once

if(game == 1)

{

//Used to fill the field up Rectangle objects

//xLocation and yLocation are both used as indexes and to show where they will be drawn

for(int xLocation=0; xLocation<10; xLocation++) //loop for columns

{

for(int yLocation=0; yLocation<10; yLocation++) //loop for rows

{

rectangle[yLocation][xLocation] = new Rectangle(xLocation\*80, yLocation\*70+100);

rectangle[yLocation][xLocation].drawRect();

}

}

for(int a = 0; a < 60; a++)

rectangle[int(random(10))][int(random(10))].diceRoll(int(random(6)+1));

game ++;

}

// if the game ends, display the end game screen

if(gameEnd)

{

background(0);

fill(255);

textAlign(CENTER);

if(oneScore > twoScore)

text("Player 1 Won!", 400, 300);

else

text("Player 2 Won!", 400, 300);

text("Player 1: " + oneScore + "\nPlayer 2: " + twoScore, 400, 350);

}

}

//Show the instructions on the screen. Stops running when the screen is clicked.

void instructions()

{

while(game == 0)

{

background(0);

stroke(255);

strokeWeight(3);

line(0,95,width,95); //Create a line to seperate the banner and the field

textSize(30);

text("How To Play:\nPlayers take turns selecting a rectangle." +

"\n60 of these rectangles have a dice hidden behind them." +

"\nThe first player to each a score above 20 wins." +

"\nClick anywhere on the screen to begin.", CENTER,150);

//Display the players' score and color code player 1 since player 1 always go first

fill(255,0,0);

text("Player 1: ", width\*.25-100, height - 740);

fill(255);

text(oneScore, 240, 60);

text("Player 2: " + twoScore, width\*.75-100, height - 740);

stroke(0);

textSize(30);

break;

}

}

class Rectangle

{

int xAxis;

int yAxis;

int dice;

//ratio of window size to individual square on the grid

int colScale = (width)/10; //Equals 80

int rowScale = (height-100)/10; //Equals 70

//Constructor to set the rectangle's X and Y axis

Rectangle(int x, int y)

{

xAxis = x;

yAxis = y;

}

void diceRoll(int d)

{

dice = d;

}

//draw the rectangle object

void drawRect()

{

rect(xAxis, yAxis, colScale, rowScale,12);

}

// turn this rectangle to black

// change the dice value to -1 to avoid being clicked twice

// return the original dice value

int clickRect()

{

int roll = dice;

dice = -1;

fill(0);

rect(xAxis, yAxis, colScale, rowScale, 12);

return roll;

}

}

void mouseClicked()

{

game++; //Used to start the actual game

// if any player's score reached 20, end the game

if(oneScore >= 20 || twoScore >= 20)

gameEnd = true;

if(gameEnd)

return;

if(game>1)

{

if(mouseY > 100)

{

// calculate the index of the rectangle based on where the player clicks

int x = mouseX/80;

int y = (mouseY-100)/70;

int roll = rectangle[y][x].clickRect();

// if roll == -1, it means that this rectangle had already been clicked

if(roll == -1)

return;

// update the score of the player

// the rect() is for erasing the previous score

if(one)

{

oneScore += roll;

rect(225, 30, 50, 50);

fill(255);

text(oneScore, 240, 60);

fill(255);

text("Player 1: ", width\*.25-100, height - 740);

fill(0,0,255);

text("Player 2: ", width\*.75-100, height - 740);

one = false;

}

else

{

twoScore += roll;

rect(625, 30, 50, 50);

fill(255);

text(twoScore, 640, 60);

fill(255,0,0);

text("Player 1: ", width\*.25-100, height - 740);

fill(255);

text("Player 2: ", width\*.75-100, height - 740);

one = true;

}

}

}

}

void mouseClicked()

{

game++; //Used to start the actual game

// if any player's score reached 20, end the game

if(oneScore >= 20 || twoScore >= 20)

gameEnd = true;

if(gameEnd)

return;

if(game>1)

{

if(mouseY > 100)

{

// calculate the index of the rectangle based on where the player clicks

int x = mouseX/80;

int y = (mouseY-100)/70;

int roll = rectangle[y][x].clickRect();

// if roll == -1, it means that this rectangle had already been clicked

if(roll == -1)

return;

// update the score of the player

// the rect() is for erasing the previous score

if(one)

{

oneScore += roll;

rect(225, 30, 50, 50);

fill(255);

text(oneScore, 240, 60);

fill(255);

text("Player 1: ", width\*.25-100, height - 740);

fill(0,0,255);

text("Player 2: ", width\*.75-100, height - 740);

one = false;

}

else

{

twoScore += roll;

rect(625, 30, 50, 50);

fill(255);

text(twoScore, 640, 60);

fill(255,0,0);

text("Player 1: ", width\*.25-100, height - 740);

fill(255);

text("Player 2: ", width\*.75-100, height - 740);

one = true;

}

}

}

}

------------------------------------------------------------------------------------------------------------------------

//Patrick - I turned Stephany’s code into an array of objects

Rectangle [] rectangle = new Rectangle[100];

int rectIndex=0; //The index of the rectangle object

void setup()

{

size(800,800);

background(0);

stroke(255);

strokeWeight(3);

line(0,95,width,95); //Create a line to seperate the banner and the field

textSize(30);

//Display the players' score

text("Player 1:", width\*.25-100, height - 740);

text("Player 2:", width\*.75-100, height - 740);

stroke(0);

//Used to fill the field up Rectangle objects

for(int xLocation=0; xLocation<10; xLocation++) //loop for columns

for(int yLocation=0; yLocation<10; yLocation++) //loop for rows

{

rectangle[rectIndex] = new Rectangle(xLocation\*80,yLocation\*70+100);

rectangle[rectIndex].drawRect();

rectIndex++;

}

}

void draw()

{

}

class Rectangle

{

int xAxis;

int yAxis;

//ratio of window size to individual square on the grid

int colScale = (width)/10; //Equals 80

int rowScale = (height-100)/10; //Equals 70

//Constructor to set the rectangle's X and Y axis

Rectangle(int x, int y)

{

xAxis = x;

yAxis = y;

}

//draw the rectangle object

void drawRect()

{

rect(xAxis, yAxis, colScale,rowScale,12);

}

}

------------------------------------------------------------------------------------------------------------

//Stephany’s code

/\*beta code for drawing grid (space left on top for the addition of the score banner)\*/

int rows;

int cols;

//ratio of window size to individual square on the grid

//rowScale = (height - 100)/number of rows

//colScale = (width)/number of cols

int rowScale = 70;

int colScale = 80;

void setup()

{

size(800,800);

background(0);

rows = (height-100)/rowScale;//10 rows

cols = (width)/colScale;//10 columns

}

void drawGrid()

{

for(int i=0; i < cols; i++)//loop for columns

{

for(int j=0; j < rows; j++)//loop for rows

{

//Preparing to draw our rectangles at coordinates(x,y)

int x = i\*colScale;

int y = j\*rowScale+100;//first rectangle is at y=100 to leave space

//at the top of the window for our score banner

fill(255);

stroke(0);

//for each rectangle drawn at (x,y), width and height are scaled by

//our pre-set colScale and rowScale to assure they fit within the window

rect(x,y,colScale,rowScale,12);//last parameter indicates the curvature of the edges

}

}

}

void draw()

{

drawGrid();

}

-----------------------------------------------------------------------------------------------------------

//Alina’s code

float xGridLine = 50, yGridLine = 50;

void setup()

{

size(400,300); //must be big enough to store 100 square shapes

//DRAWS GRID

background(0);

drawGridLine(xGridLine, width, height);

drawGridLine(yGridLine, height, width);

//END OF GRID

}

void draw()

{

}

void drawGridLine(float numLines, float amnt, float length) {

stroke(255); //makes lines white

while(numLines < amnt) {

line(numLines, 0, numLines, length);

numLines = numLines +50;

}

}

Each group should do the following with the game assigned to them:

**1. Is this game considered a “game” according to the definitions we covered in class? Discuss. If the game is**

**a game according to some definitions but not all, explain both.**

**According to Jesse Schell, “A game is a problem solving activity approached with a playful attitude”. Our game with this definition is not a game because it is based on luck. You aren’t solving any problems, unless Schell considers one’s score being zero a problem. In that case, It is a problem solving activity where the player is trying to get a score of 20 before the other player. Since Schell deems rock paper scissors, which is also based on luck and not a problem solving activity as a game, it is safe to say our game would be considered a game by Jesse Schell.**

**By Jane McGonigal’s definition, our game is a game because there is a goal (to get to 20), rules (you click on a square to find your score), feedback system (keep tracks of score, and when you win or lose), and voluntary participation.**

**Maroney says “A game is a form of play with goals and structure”. There is structure because a player can click on a square and either lose points and suffer the consequences of increasing them, or gain points and get closer to 20. There is a goal, to get to score of 20 and higher. Therefore, our game can be considered a game.**

**Lastly, by Crawford’s definition our game is not a game because the players are not directly interacting in such a way that impedes each other’s goal. Instead, our game would be considered a challenge.**

**2. What makes this game seem “broken”? Why is it not very much fun to play?  
The game seems broken because it lacks all the eight elements of fun. The fun part about rolling dice can be sensation and submission, but digitally that is gone. There is also not enough surprises (besides the one factor where you don’t know who will win), which is important to a game.**

**3. What are the constitutive rules of the game?**

**The constitutive rules for the given game is that two players take alternate turns to roll a dice, and whoever gets up to 20 first wins.**

4. How can you make this game more fun? Brainstorm, come up with a revised game, and playtest. Then

brainstorm and revise more as necessary and playtest again!

**5. What are the operational rules of your new game? What are the constitutive rules? Which elements of fun (of the eight that we discussed) did you incorporate?  
The constitutive rules are that hidden under a grid of shapes, there are 3 different items: a blank, a dice to roll for positive points, and a trap that cuts points off your score. Two players take alternate turns to pick a shape, and reveal whether their score increased, decreased, or had no effect. The first player that gets to 20 or above wins.**

**The operational rules is that the shapes are laid out as a 6\*10 grid of squares. There is a random number of traps, and a random number of dice for each game, where the dice is always more than the number of traps. Either player can press “R” to start the game, and the game starts off with player 1 going first.**

**From the eight elements of fun, we incorporated challenge, and fellowship:**

* **We attempted to make the game more challenging by creating traps and blanks so that it’s more difficult to get to 20. We also added a variable number of dice and traps to keep the player guessing on the probability of winning. We wanted to add surprise by hiding the outcome under shapes, and so that the player has less control over their outcome. They can uncover either a loss, a gain, or nothing.**
* **Fellowship was already a part of the two player game, but with the added challenges, fellowship between two players is prolonged due to friendly competition and increased longevity of the game.**

**Brain Storming:**

**While brainstorming, we thought about ways to make the game more fun through added challenges. With the game battleship in mind, we chose a grid layout to hide our outcomes under - similar to hiding boats in battleship. Then, the player clicks a spot on the grid to reveal the outcome. Originally, we implemented a 10\*10 grid, but in practice this was too long and their was too many squares left un-clicked. Also, it made the user interface look awkward so we picked a 6\*10 grid.**

**Next, originally instead of traps we thought of a gamble a player can take twice throughout the course of the game. Both gambles resulted in two outcomes: either doubling the player’s current number of points or loosing half the amount of current points. We decided to go with traps instead because it made the game too easy to win when doubling points. The traps create more risk.**

**If we had more time, we would highlight a 3\*3 spot on the grid hinting that somewhere in this area, there is a dice, as opposed to a trap or a blank. The area would be highlighted only for one player and they get two chances (two clicks). This bonus is awarded if they get a score of 10 or 15 before the other player. After two tries, the grid switches up to different outcomes and the other player makes their move like regular. We believe this will create more risk, which can be exciting. There is a higher probability of getting a dice with a hint, so not getting a dice can be more frustrating, which increases the challenge.**

6. Create a digital version of your game using Processing. You may assume that two (or more) players are

standing around one computer sharing the monitor and keyboard. Make sure that each player knows when

it is his/her turn. Playtest again after you digitize it! Note: it is highly recommended that you come up

with a working version of your game before programming.

It is not necessary that your revised game have the same operational medium as the original. For example,

you do not have to reproduce the complex game board of Candy Land; obviously, you will not have dice or

coins in your digitized game. You may revise starting with the constitutive rules of your given game.

Your submission should consist of the following:

• A PDF document (no other file formats will be accepted) with written answers to questions 1 – 3 and 5.

The document should also contain a brief description of your brainstorming/creation process and insights

gleaned from playtesting (e.g. “Initially, we planned to do X but when we played it, we realized that Y

would be better . . . ”)

• Your digital game posted on OpenProcessing and a file / zipped folder of your code. Make sure that the

directions are included!

Name the PDF file GroupNumberProject1 (e.g. Group6Project1), and make sure it includes each of your names.

Name your code files GroupNumberProject1 and make sure all of your names are included in the comments.

Your submission should be submitted as a group submission on Blackboard. Note that only one member of the

group need submit (but all of you will be responsible if it is late or missing). Your submission should include the

PDF file, the URL of your game on OpenProcessing and the source code.

You will also each complete an evaluation of every member of your group (including yourself!). The group

evaluation sheet is attached to this assignment as well. There is a separate submission link on Blackboard for the

group evaluation.

This assignment is due on Tuesday, October 30 at 11:59 PM.

This assignment is worth 17 points:

• 5 points for the written portion

• 12 points for the game. When grading, I will look to see functionality (able to be played, bug-free),

satisfaction of the requirements, creativity, and an aesthetically pleasing design.

• Group evaluations will be used in calculating the final grade for each individual group member.