Mancala

By Christopher SanGiovanni and Brandon Goldsmith

What is Mancala

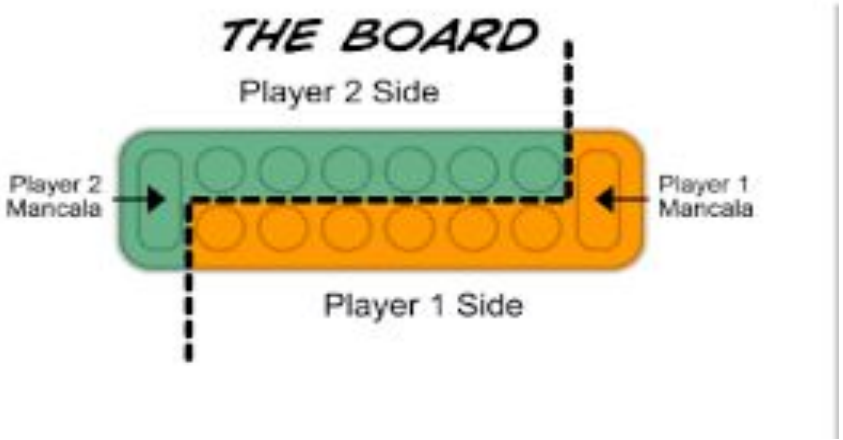
Mancala is a classic game in which players compete for to capture all the marbles in their respective goals.

Rules of the Game

Each Player has 4 marbles in each cup. The starting player will choose a cup from their side, and pick up all of its marbles.

Then, one marble is dropped in the cups until the player runs out of marbles, dropping a marble in their own goal if they pass it.

If a player lands on their own goal, it's their turn again.



How we designed the GUI

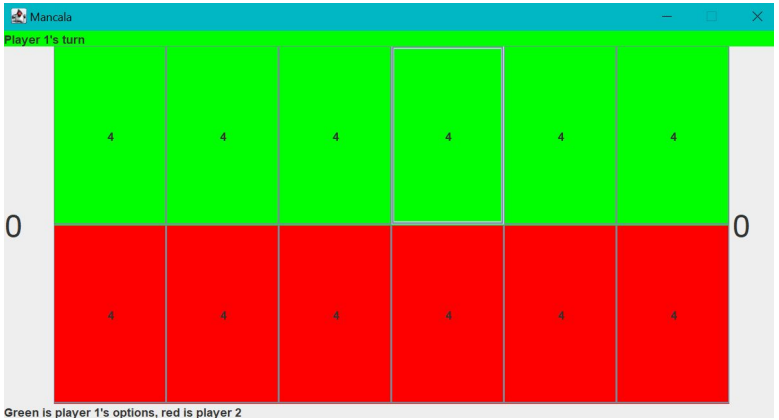
-We opted to use two main components(JPanels) for our GUI, this way we could utilize nested layouts.

-The BorderLayout option allowed for Top and bottom labels, as well as the players' goals.

-The GridLayout option is used in the gameplay panel. This way, we could organize our buttons like a 2D array or grid.

-With the components properly organized, we

could style our game as we see fit.



GUI Layout Explanation

```
75  public void initializeButtons() {
76      JPanel buttons = new JPanel(new GridLayout(2, 6));
77      int w, h;
78      for (int i = 0; i < 2; i++) {
79          if (i == 1) {
80              for (int j = 0; j < 6; j++) {
81                  w = this.getWidth();
82                  h = this.getHeight();
83                  cups[i][j] = new JButton();
84                  cups[i][j].setPreferredSize(new Dimension(75, 75));
85                  cups[i][j].setFocusable(false);
86                  if (i == 0)
87                      cups[i][j].setBackground(Color.GREEN);
88                  else
89                      cups[i][j].setBackground(Color.RED);
90                  buttons.add(cups[i][j]);
91              }
92          } else if (i == 0) {
93              for (int j = 5; j >= 0; j--) {
94                  w = this.getWidth();
95                  h = this.getHeight();
96                  cups[i][j] = new JButton();
97                  cups[i][j].setPreferredSize(new Dimension(75, 75));
98                  cups[i][j].setFocusable(false);
99                  if (i == 0)
100                     cups[i][j].setBackground(Color.GREEN);
101                 else
```

Man Frame

We started with a class called ManFrame (extends JFrame), which creates our frame, and adds a Manpanel (extends ManPanel).

```
class ManFrame extends JFrame {
    ManFrame() {
        ManPanel panel = new ManPanel();
        this.setTitle("Mancala");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);
        this.add(panel);
        this.pack();
        this.setResizable(false);
        this.setVisible(true);
    }
}
```

Man Panel

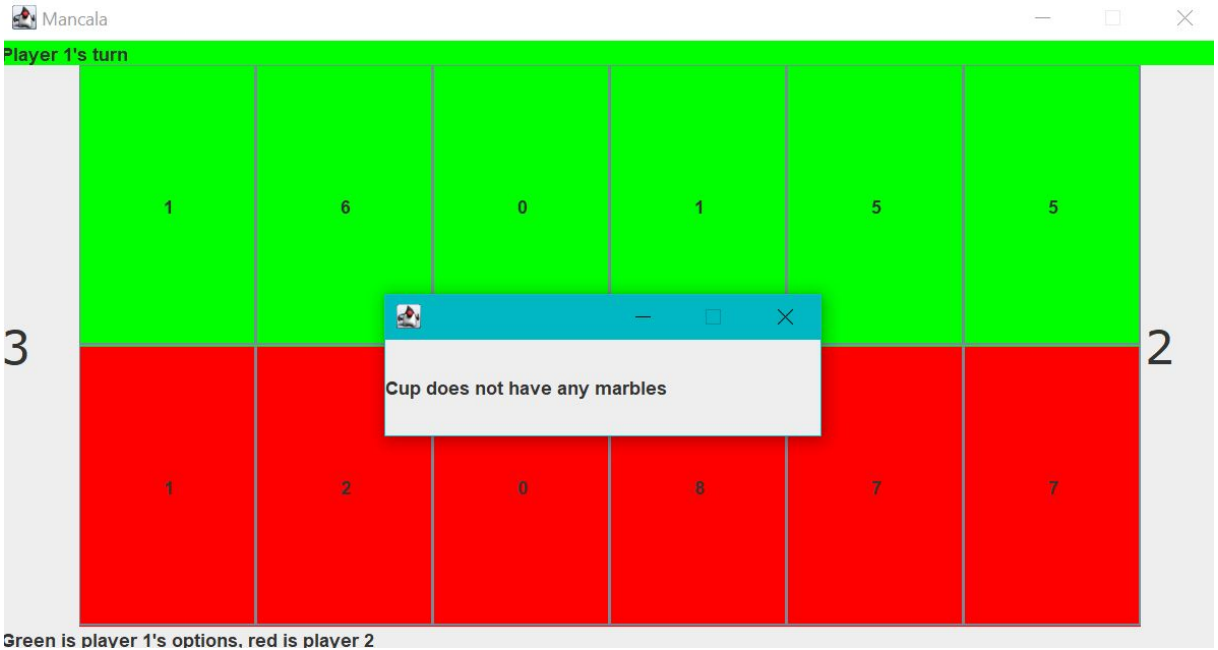
```
9   class ManPanel extends JPanel
10  {
11      private Integer[][] boardValues = new Integer[2][6];    //integer array that holds all of our values of each cup
12      private Integer leftGoal;               //integer for left goal
13      private Integer rightGoal;              //integer for our right goal
14      private int turn;                       //determines the player's turn (odd is player 2, even is player 1)
15      private JButton[][] cups = new JButton[2][6];          //button array for all of the places on our board
16      ManPanel(){
17          //initializes all of our variables
18      }
19      public void initializeButtons(){
20          //initializes all of our buttons and organizes them in our grid layout
21      }
22      public void reSetNames(){
23          //connects the
24      }
25      public void cupIsPressed(int x, int y){
26          //is called from the action listener on each button and tests the index to see if it is valid
27          //if it is the correct turn/ if it is empty/ and if the game has ended
28      }
29      public boolean endOfGameChecker(){
30          //returns a boolean and tests to see if one of the sides of the board is empty
31      }
32      public void move(int x, int y){
33          //actually moves the marbles using a while loop along with checking for capture or extra turn
34      }
35  }
```

Move()

```
32    public void move(int x, int y) {
33        int currentX = x;
34        int currentY = y + 1;
35        while (boardValues[x][y] > 0)
36        {
37            if (currentY == 6 && boardValues[x][y] != 0)
38            {
39                if (x == 0 && currentX == 0)
40                {
41                    leftGoal++;
42                    boardValues[x][y]--;
43                    currentY = 0;
44                    currentX = 1;
45                    if (boardValues[x][y] == 0)
46                    {
47                        turn--;
48                    }
49                }
50                else if (x == 0 && currentX == 1)
51                {
52                    currentY = 0;
53                    currentX = 0;
54                }
55                else if (x == 1 && currentX == 1)
56                {
57                    rightGoal++;
58                    boardValues[x][y]--;
59                    currentY = 0;
60                    currentX = 0;
61                    if (boardValues[x][y] == 0)
62                    {
63                        turn--;
64                    }
65                } else if (x == 1 && currentX == 0)
66                {
67                    currentY = 0;
68                    currentX = 1;
69                }
70            }
```

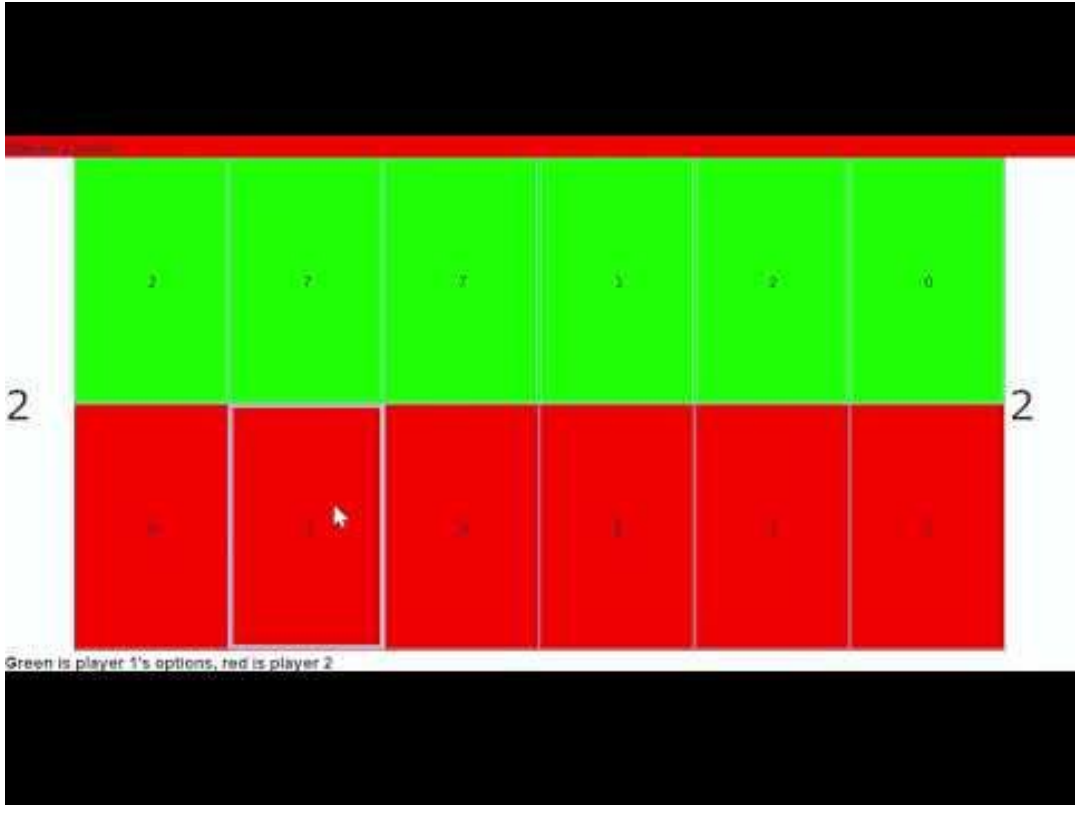
```
71            if (boardValues[x][y] != 0)
72            {
73                boardValues[x][y]--;
74                boardValues[currentX][currentY]++;
75                if (boardValues[x][y] == 0 && boardValues[currentX][currentY] == 1 && currentX == x)
76                {
77                    int oppositeX = 0, total = 0, oppositeY = 0;
78                    if (currentX == 0)
79                        oppositeX = 1;
80                    else if (currentX == 1)
81                        oppositeX = 0;
82
83                    switch (currentY) {
84                    case 0:
85                        oppositeY = 5;
86                        break;
87                    case 1:
88                        oppositeY = 4;
89                        break;
90                    case 2:
91                        oppositeY = 3;
92                        break;
93                    case 3:
94                        oppositeY = 2;
95                        break;
96                    case 4:
97                        oppositeY = 1;
98                        break;
99                    case 5:
100                       oppositeY = 0;
101                       break;
102                   }
103                   if (boardValues[oppositeX][oppositeY] != 0)
104                   {
105                       total = boardValues[oppositeX][oppositeY] + boardValues[currentX][currentY];
106                       boardValues[oppositeX][oppositeY] = 0;
107                       boardValues[currentX][currentY] = 0;
108                       if (turn % 2 == 0) {
109                           leftGoal += total;
110                       } else if (turn % 2 == 1) {
111                           rightGoal += total;
112                       }
113                   }
114               }
115               currentY++;
116           }
```

Error: empty cup



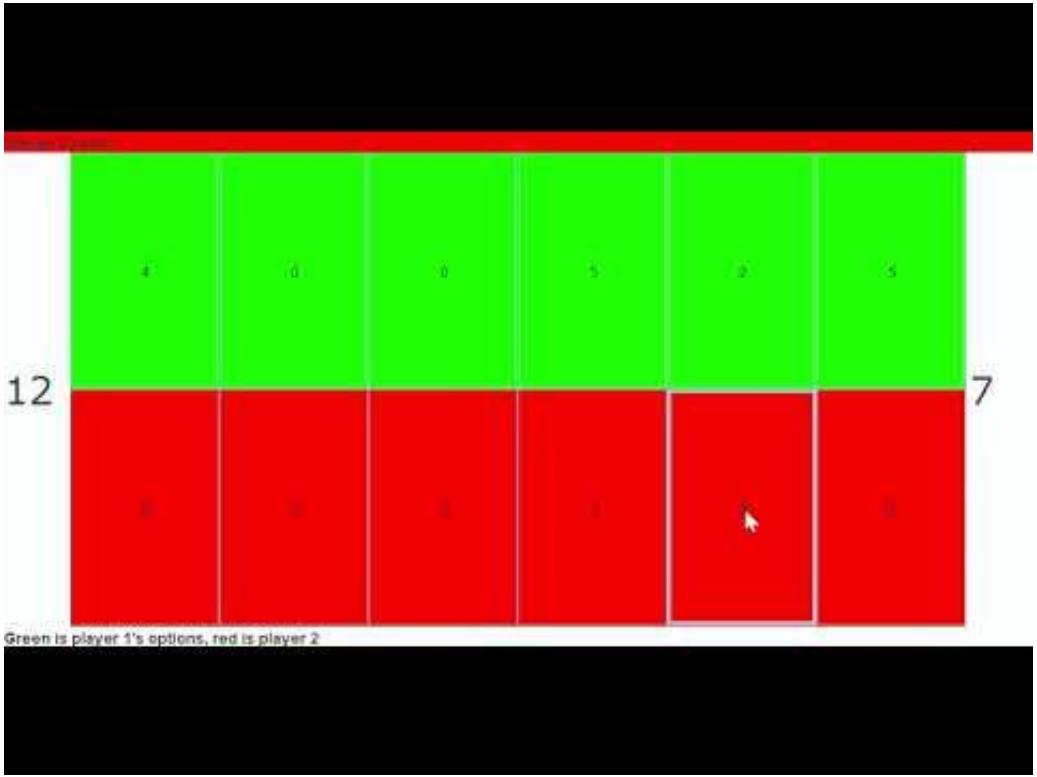
Error: Not your turn

Capture



[Capture Video]

Extra Turn

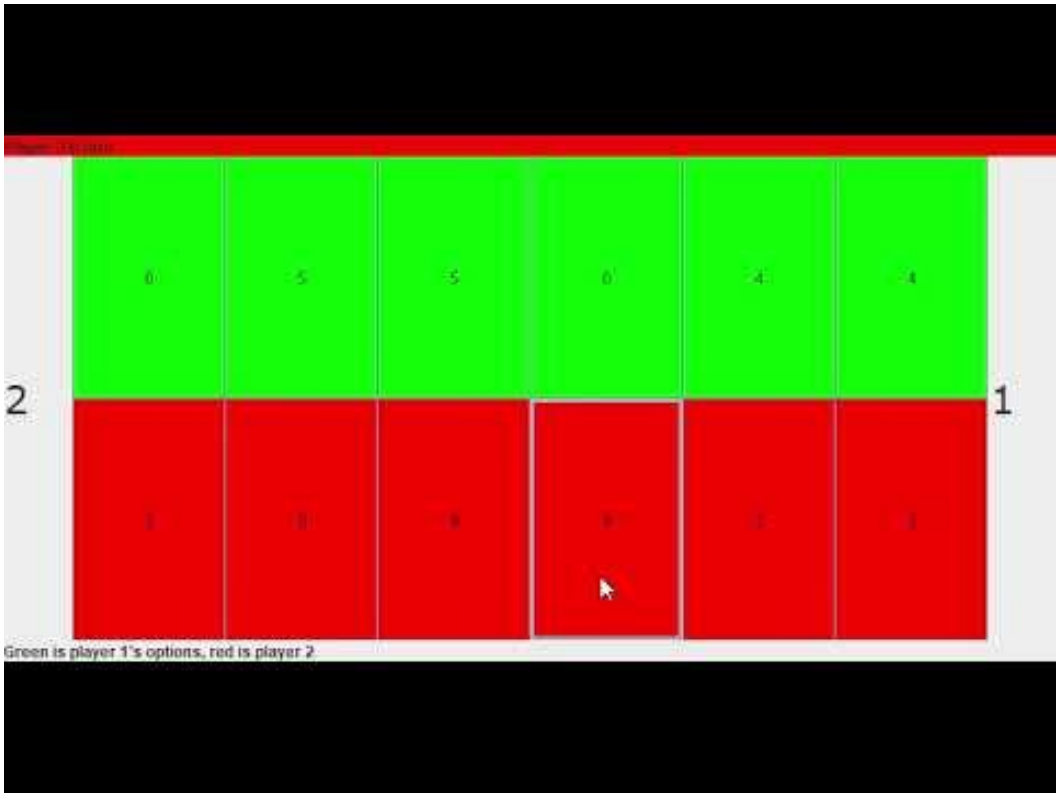


Extra Turn video

Winner


Mancala
Player 1's turn
0
0
0
0
0
0
13
35
Player 2 wins!
0
0
0
0
0
0
Green is player 1's options, red is player 2

Example/Demo



Gameplay