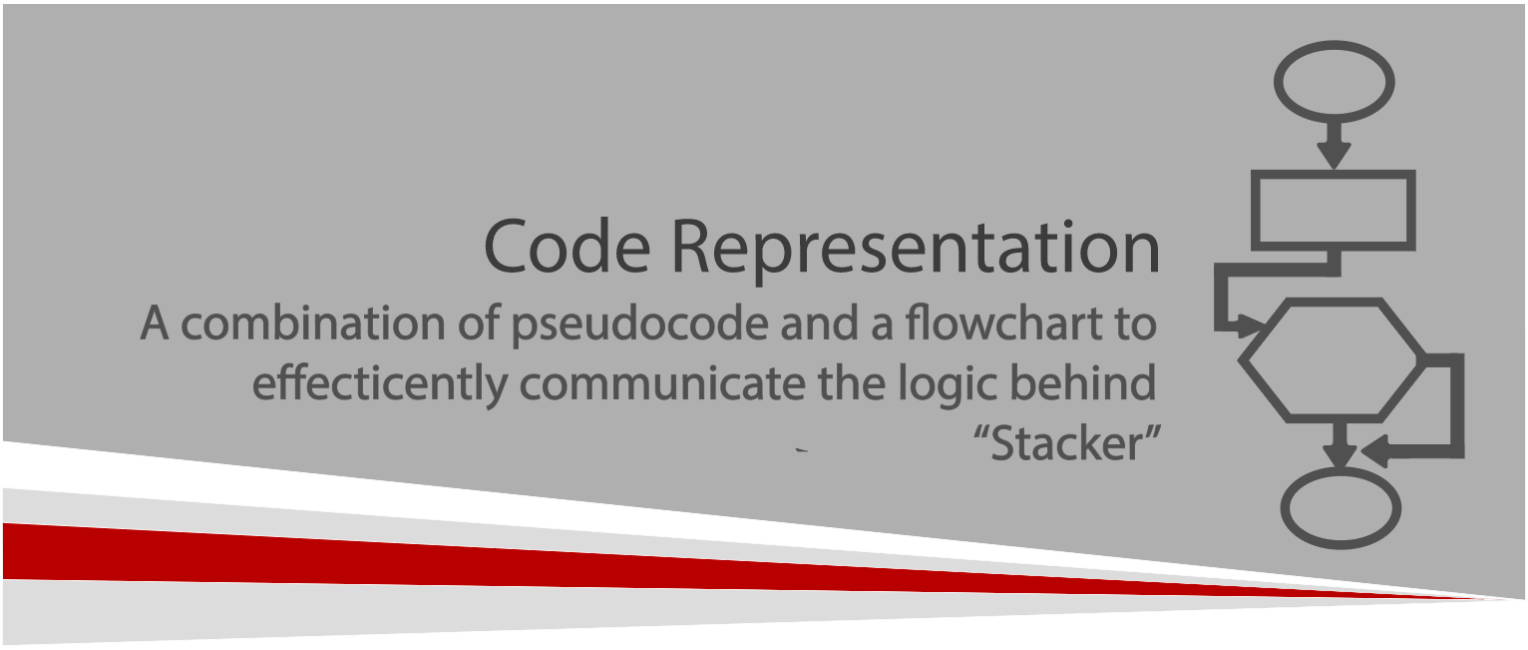
Software Documentation

This section contains the software documentation for the design project. This includes an introduction to the project, detailed discussion of the code, and the results.

- Program Description for Developers
- Final Code



Pseudocode for overall game

Start: Display “Welcome to game. Please touch screen to begin”

if user selects “Play the Game”

Reference subprogram flowchart found here

if user selects “Rules/Instructions”

Print to screen:

“Stacker Rules:

The goal of Stacker is to stack a tower of blocks as high as possible.

To place a block, tap the screen. The higher you stack, the more points you get.

But be careful, if you miss the stack, it is GAME OVER!”

Wait for an additional screen touch to return to the main menu

if user selects “Statistics”

Opens “scores.txt”, a text file with the game top 5 high scores listed

Reads in the scores to name and score arrays

Declare counter i, for i = 0; i < 5; i++

Check if the player's score is greater than score[i]

If so, assign the player's score to score[i]

Assign each subsequent value of the score vector to its i+1 value (score[2] becomes score[3])

Print the high scores table to the proteus screen

Write the updated high score names and scores to "scores.txt"

Close the file.

Wait for an additional screen touch to return to the main menu

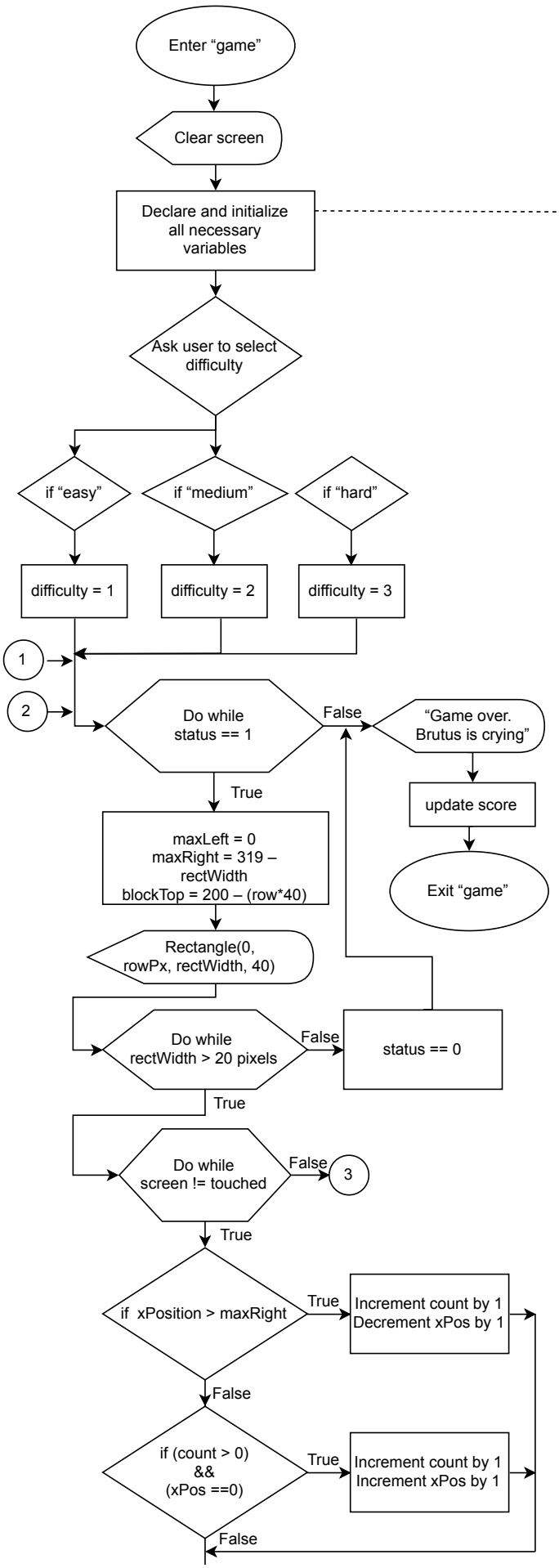
if user selects "Credits"

Print to screen:

> "Creators: Annie Bete and [redacted]. A big thank you to FEH instructors [redacted] for helping with project design and troubleshooting. Also, thank you to the FEH proteus resources found out https://u.osu.edu/fehproteus"

Wait for an additional screen touch to return to the main menu

Flowchart for Game



Assumptions

All necessary variables are declared and initialized

Initial declarations

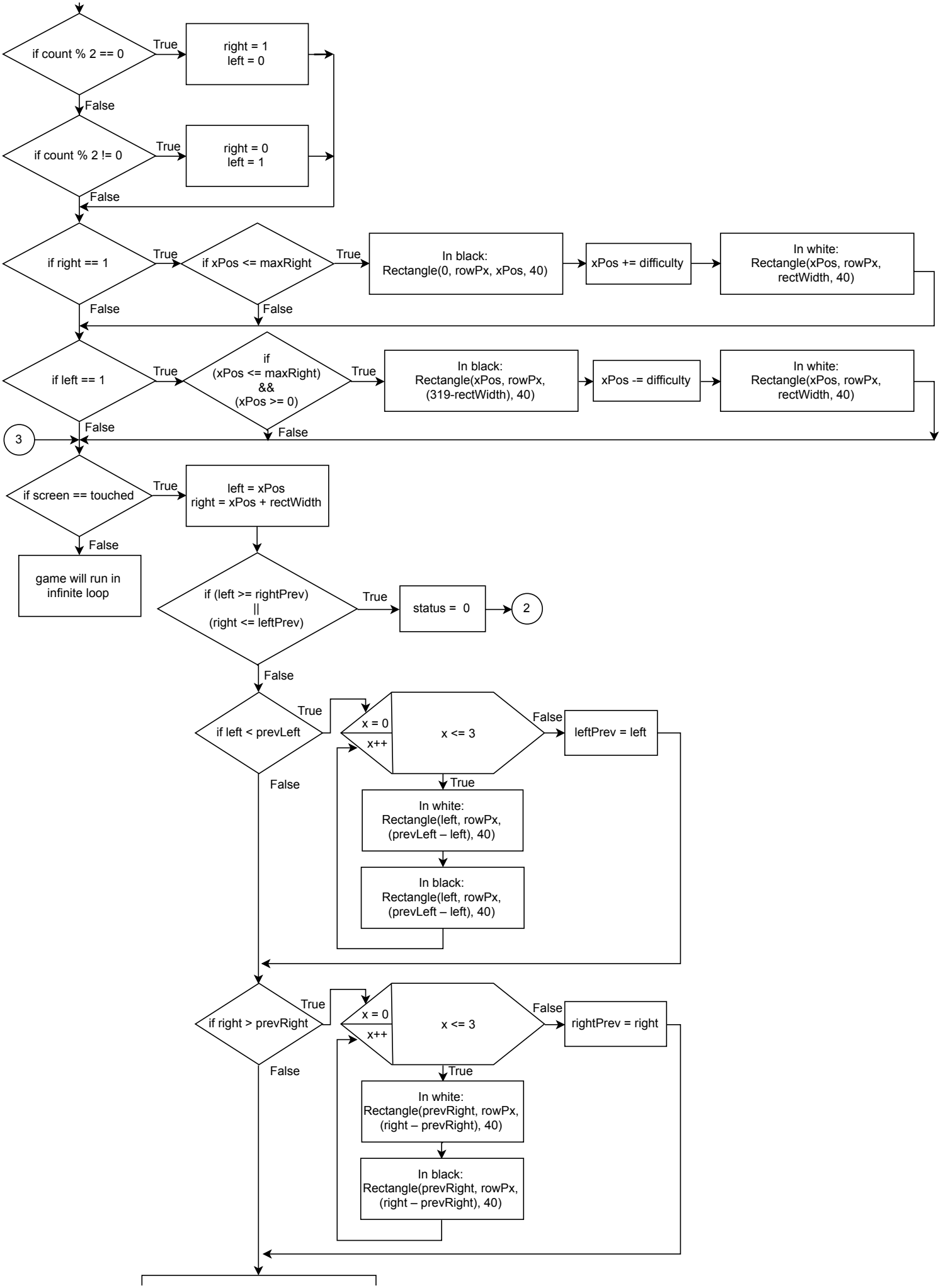
For select variables:

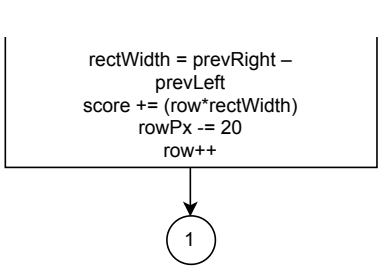
count = 0
status = 1
rectWidth = 200
xPos = 0
row = 1
rowPx = 200
left = xPos
right = xPos + rectWidth
prevLeft = 0
prevRight = 319
score = 0

Notes on Drawing a Rectangle

Rectangle(a, b, c, d)

draws a rectangle with width “c” and height “d” where its top left corner is at x-pixel “a” and y-pixel “b”


if count % 2 == 0
True
right = 1
left = 0
False
if count % 2 != 0
True
right = 0
left = 1
False
if right == 1
True
if xPos <= maxRight
True
In black:
Rectangle(0, rowPx, xPos, 40)
xPos += difficulty
In white:
Rectangle(xPos, rowPx,
rectWidth, 40)
False
False
if left == 1
True
if
(xPos <= maxRight)
&&
(xPos >= 0)
True
In black:
Rectangle(xPos, rowPx,
(319-rectWidth), 40)
xPos -= difficulty
In white:
Rectangle(xPos, rowPx,
rectWidth, 40)
False
False
3
if screen == touched
True
left = xPos
right = xPos + rectWidth
False
game will run in
infinite loop
if (left >= rightPrev)
||
(right <= leftPrev)
True
status = 0
2
False
if left < prevLeft
True
x = 0
x++
x <= 3
False
leftPrev = left
True
False
In white:
Rectangle(left, rowPx,
(prevLeft – left), 40)
In black:
Rectangle(left, rowPx,
(prevLeft – left), 40)
if right > prevRight
True
x = 0
x++
x <= 3
False
rightPrev = right
True
False
In white:
Rectangle(prevRight, rowPx,
(right – prevRight), 40)
In black:
Rectangle(prevRight, rowPx,
(right – prevRight), 40)


rectWidth = prevRight –
prevLeft
score += (row*rectWidth)
rowPx -= 20
row++
1

Assumptions

- Developers have read over the User Manual and Program Features.
- Developers are familiar with C/C++, along with the Proteus.

Important Variables Names and Uses

- **Variables for main game functionality:**
  - **status** – indicates if the game loop should continue. Default value is 1, but will be 0 if the rectangle's width is under its minimum 20 pixel size, or if the user does not land any of the block on the previous row.
  - **score** – keeps track of user's score. Calculated by multiplying the row number (plus one) by the number of pixels from current row's rectangle that landed on the previous row.
  - **difficulty** – selected from user input, it determines the speed at which the rectangle moves.
- **Variables to determine if the block's movement:**
  - **count** – incremented after the rectangle has moved its maximum distance to the right. It is modulated by two to determine if the block is moving left or right. If count % 2 equals 0, the block is moving to the right and if count % 2 does not equal two, the block is moving to the left.
  - **xPos** – the pixel value of the left most x-value of the rectangle
  - **blockTop** – the pixel value of the top most y-value of the rectangle. It uses the row number to calculate the position the rectangle should be on the screen.
- **Variables to determine what portion of the block lands on top of the previous row:**
  - **left** – indicates the current leftmost x-value for the current row's rectangle
  - **right** – indicates the current rightmost x-value for the current row's rectangle
  - **prevLeft** – indicates the leftmost x-value from the previous row
  - **prevRight** – indicates the rightmost x-position value from the previous row

Classes

- **class Game** – used to keep track of each user's stats. New class is instantiated at the beginning of each game.
  - **Private variables:** name, score, count, difficulty
  - **Public:** function game() and constructor Game()

Function Descriptions

- **void game() –** flowchart for function can be found here. This function contains all of the code for the loop that brings the user through choosing their difficulty level and playing the game. The function finishes when the user looses (variable status becomes 0) and the score for that user is updated.
- **void dispRules()** – prints all of the stacker rules to the screen when function is called.
- **int main()** contains:
  - A **welcome screen** that user taps to transition to the **main menu**, where user can choose to "play game", "view stats", "view rules", or "view acknowledgments".
  - **Play game** – instantiates the class "Game", calls function "game()", and allows user to tap screen to return to main menu.
  - **Stats** – contains all of the code that reads and writes top scores to the SD card and prints to screen. Allows user to tap screen to return to main menu.
  - **Rules** – calls function "dispRules()" and allows user to tap screen to return to main menu.
  - **Acknowledgements** – prints all acknowledgements to screen and allows user to tap screen to return to main menu.

Program Performance

- The program is not fully performing. It does not exit the game function when selected, although the majority of the functionality (as described on the Features Page) did work.

Limitations

- The game has a maximum height of six rows.