Brandon Atwal BSA190001

Professor Mazidi

CS 2240

April 25, 2021

Get Fat Bitmap Project

**Run Instructions**

First you must open the Bitmap Display and “Keyboard and MMIO Simulator” (keyboard). On the Bitmap Display, set the pixel dim to 4x4, then set the display dim to 256x256. Be sure to resize the window so the whole display is visible. Use $gp as the base address. Then press the button on both windows to connect them to MIPS. Once that is done, you may build the program, then run it.

The program should now be running. Once you’ve admired the title screen, press space to start the game. You can control the cursor through the Keyboard and Display MMIO Simulator. W, A, S, D is used to move the blob. Space is used to quit the game.

**Written Overview**

The program is called Get Fat, a game where you play a hungry blob that wants to become giant. When the player begins the game and passes the title screen, they will see a rainbow-colored blob and several colored dots on the screen. The player’s blob can “eat” dots by running into them. Blue dots are food, and can be eaten to make the blob larger. The blob will become larger after a certain amount of food, dependent on its size. The larger the blob becomes, the more food it needs to become larger. Green dots are poison, and the blob will become smaller if it eats it. If the blob is very small and it eats poison, it will die, resulting in a game over. Magenta dots are muck, and the blob will become slower if it eats it. Yellow dots are grease, and the blob will become faster if it eats it. Grease can be used to counter-act the effects of sap. If the blob dies, or the player exits via the space bar, a game over screen appears.

**Pseudocode**

Word does weird things to pseudocode. If needed, I can provide the original .txt file.

Pseudocode for Get Fat by Brandon Atwal

main{

titleScreen{

call blackAll

call printTitle

titleLoop{

if the user presses space, continue to start

else keep waiting for user input

}

}

start{ //intialize the program

call blackAll //to erase the title screen

initialRandomize{

randomize locations of all dots

}

paint every dot to bitmap display

initialize the blob and put it at its starting location (center of screen){

X = width/2

Y = height/2

color = RED

}

}

gameLoop{

drawBlob{

for (int i = 0; i < width; i++){

X++;

call changeColor

draw pixel

wait(waitTime)

}

for (int i = 0; i < width; i++){

Y++;

call changeColor

draw pixel

wait(waitTime)

}

for (int i = 0; i < width; i++){

X--;

call changeColor

draw pixel

wait(waitTime)

}

for (int i = 0; i < width; i++){

Y--;

call changeColor

draw pixel

wait(waitTime)

}

}

processDots{

call checkDots

save X, Y and color in temporary registers

get food1 location

set the color to blue

draw pixel

get food2 location

set the color to blue

draw pixel

get poison1 location

set the color to green

draw pixel

get muck1 location

set the color to magenta

draw pixel

get muck2 location

set the color to magenta

draw pixel

get grease1 location

set the color to yellow

draw pixel

}

getInput{

if (keyboard/MMIO simulator has input){

while (input not valid){

loop until valid input found

}

if (input is 32){ //if input is space key

jump to exit

}

else if (input is 'w'){

if (Y > 0){

call eraseBox

Y--;

}

}

else if (input is 's'){

if (Y < (63-width)){

call eraseBox

Y++;

}

}

else if (input is 'a'){

if (X > 0){

call eraseBox

X--;

}

}

else if (input is 'd'){

if (X < (63-width)){

call eraseBox

X++;

}

}

}

}

processGrowth{

if(foodTillGrow <= 0){

call eraseBox

timesGrown++

width++

if (width==7 || width==14){

width++

}

if (width>=7){

foodTillGrow = 8

}

else if (width >=){

foodTillGrow = 5

}

else{

foodTillGrow = 3

}

}

}

processShrink{

if (widthToInsert > 0){

width = widthToInsert

widthToInsert = 0

}

}

}

exit{

call blackAll

call paintGameOver

printString("Game Over")

end program

}

drawPixel{

address = $gp + 4\*(x + y\*width)

draw pixel at address

}

checkDots{

find top left coordinate and bottom right coordinate of blob

use those to detemine

if (location of food1 dot is not outside of blob){

foodTillGrow--;

randomize coordinates of this dot

}

if (location of food2 dot is not outside of blob){

foodTillGrow--;

randomize coordinates of this dot

}

if (location of poison1 dot is not outside of blob){

if (width <= 1){

call exit

}else{

widthToInsert++;

}

randomize coordinates of this dot

}

if (location of muck1 dot is not outside of blob){

if (waitTime<12){

waitTime++;

}

randomize coordinates of this dot

}

if (location of muck2 dot is not outside of blob){

if (waitTime<12){

waitTime++;

}

randomize coordinates of this dot

}

if (location of grease1 dot is not outside of blob){

if(waitTime > 2){

waitTime--;

}

randomize coordinates of this dot

}

}

wait(waitTime){

wait for waitTime ms

}

changeColor{

i = 0

if (color = red){

change color to first element of colors array

}else{

loop through colors array while incrementing i until the current selected color is found{

then add (i\*4) to the current array address to get next element's address

}

}

}

eraseBox{

save the old color and the return address in temporary registers

make the color black

for(int i = 0; i < width; i++){

X++;

draw pixel

}

for (int i = 0; i < width; i++){

Y++;

draw pixel

}

for (int i = 0; i < width; i++){

X--;

draw pixel

}

for (int i = 0; i < width; i++){

Y--;

draw pixel

}

restore old color and return address

}

blackAll{

while (Y != display HEIGHT-1){

while (X != display WIDTH-1){

draw pixel

X++;

}

X=0

Y++;

}

}

printTitle{

draw the title text to the display using a variety of for and while loops

}

paintGameOver{

draw the game over text and sad face to the display using a variety of for and while loops

}

}

**Sample Run Screenshots**







