Kieran Clarke

592422

CMPT 360 A

Assignment #5

2D Time Trial Racing Game (GML)

This assignment fulfills the following goals:

- an internet-downloaded language (GML) (MAC)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Assign # | Date Due | Group(s) | Language | Language | Platform |
| 1 | Sept 17 | 1 & 1 | Python | C++ | Windows (x2) |
| 2 | Oct 1 | 1 & 2 | C++ | Visual Basic | Windows (x2) |
| 3 | Oct 12 | 3 | JavaScript |  | Windows |
| 4 | Oct 24 | 1 & 4 | Pascal | Ada | Windows(x2) |
| 5 | Nov 5 | internet | GML |  | Windows |
| 6 | Nov 19 |  |  |  |  |
| 7 | Dec 3 |  |  |  |  |

We are asked to download a language from the internet, and create a game. I proposed a top-down 2D time trial racing game. The game does not have to be intelligent enough to play against the computer, so I chose time trial so that you can play against yourself and friends.

Sample input:

W is move forward (gas)

A is turn left

S is slow down (brake)

D is turn right

CTRL is reverse (only works if not moving forward)

Sample output:

movement forwards, backwards, left, right with the car

Documentation:

Use this game to race against yourself, and your friends, in a simple time-trial setting.

Error Handling:

added ways to prevent the user from cheating by reversing over the starting line.

Statement of steps required to solve the problem:

This is the code used to do so, found within obj\_car\_Step\_1:

//collision checking to see if player is cheating by going backwards over the starting line

if place\_meeting(x,y,obj\_check)

{

if place\_meeting(x,y,obj\_start)

{

room\_goto(rm\_press)

}

}

//if they are not cheating and succesfully pass the finish line, it takes them to another room

else

{

if place\_meeting(x,y,obj\_start)

{

room\_goto(rm\_done)

}

}

Pseudocode:

The code is too long to translate into pseudocode, and there are comments to explain the algorithms I created. The language is also a very readable language, and does not need to be translated into pseudocode.

This game is downloadable as a .exe at this link:

http://www.mediafire.com/file/yj194eb6ez799y5/Time\_Trial\_Racing\_Game\_-\_Drift.exe/file

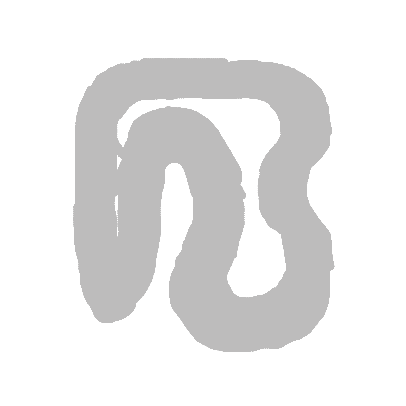
SPRITES

(I CREATED ALL THESE SPRITES IN PHOTOSHOP CS6)

spr\_240sx:



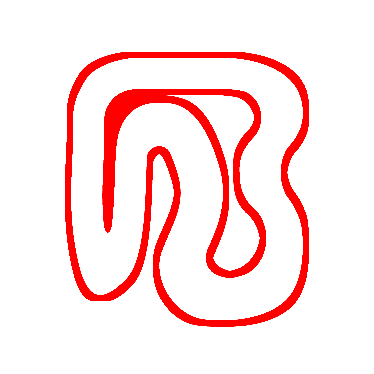
spr\_pavement:



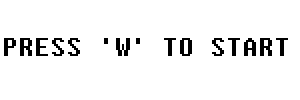
spr\_start:

(is a white rectangle)

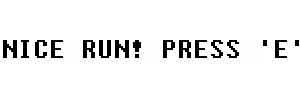
spr\_wall:



spr\_txt\_press:



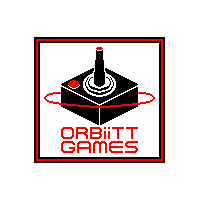
spr\_txt\_press2:



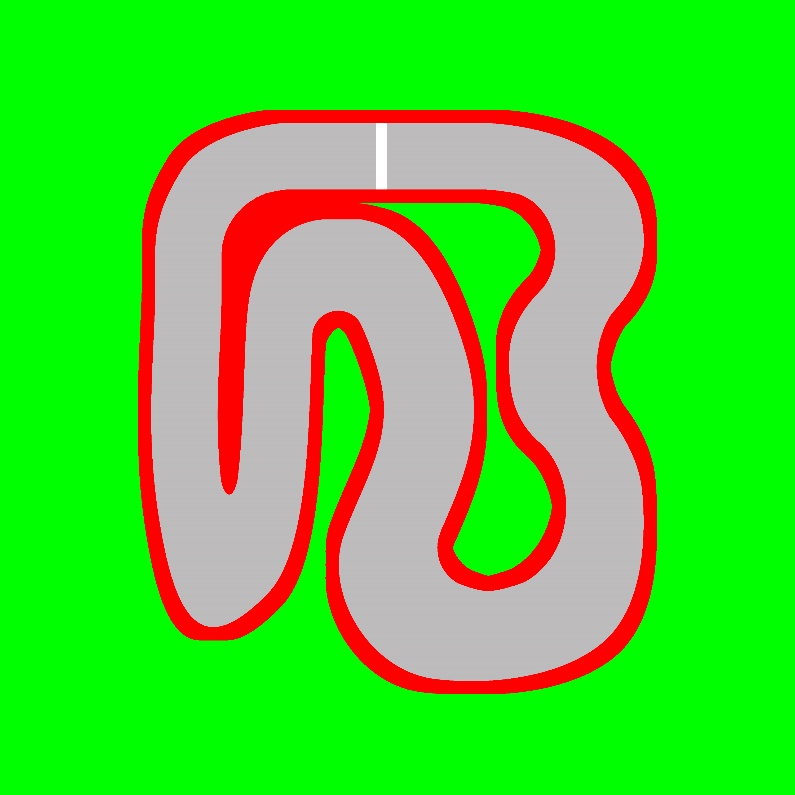
spr\_check:

(is invisible, used to check if player is cheating by going backwards over the finish line)

spr\_logo:



THE WHOLE TRACK:



SCRIPTS

scr\_loadHiScore:

if(file\_exists("high\_Score.txt")) //check if there is already a file created

{

var var\_loadFile; //create variable

var\_loadFile = file\_text\_open\_read(working\_directory + "high\_Score.txt") //read file

var\_highScoreData = file\_text\_read\_real(var\_loadFile); //gets the highscore

file\_text\_close(var\_loadFile); //close the file you are reading

}

Scr\_SaveHiScore:

//checks to make sure you are in the specific room you go to after succesfully completing a race

if room == rm\_done

{

//if your score is less than the highscore (it is time trial so you want a lesser score)

if(score < var\_highScoreData)

{

//sets the in game highscore as what is in the file

var var\_saveFile = file\_text\_open\_write(working\_directory + "high\_Score.txt");

file\_text\_write\_real(var\_saveFile,score);

//close the file

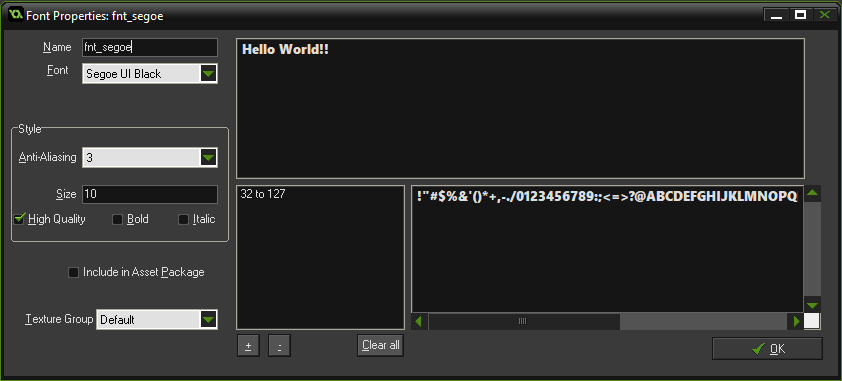
file\_text\_close(var\_saveFile);

}

}

Fonts

Fnt\_segoe:



OBJECTS

obj\_car:

(uses spr\_240sx)

obj\_car\_Create\_1:

depth = -10; //for layer order

forSpd = 1; //forward speed

forSpdAcc = 2.88; //adjust between 2.85 and 2.9 for diff max speed & acceleration

backSpd = -0.1; //backwards speed

backSpdAcc = 1.2; //backwards acceleration

brakeDecel = 0; //deceleration rate from braking starts at 0

turnAccL = 0; //smooth turning acceleration LEFT starts at 0

turnAccR = 0; //smooth turning acceleration RIGHT starts at 0

/\*

I did not fully figure out the drifting mechanics. I was able to create code for

the car to smoothly enter a drift, but could not quite figure out how to smoothly

exit the drift. so I left this part out of the game.

\*/

drDirL = 0;

drDirR = 0;

drAngL = 0;

drAngR = 0;

obj\_car\_Step\_1:

//keybinds

left = keyboard\_check(ord("A"));

right = keyboard\_check(ord("D"));

down = keyboard\_check(vk\_control);

up = keyboard\_check(ord("W"));

eb = keyboard\_check(vk\_space);

brake = keyboard\_check(ord("S"));

//collision checking to see if player is cheating by going backwards over the starting line

if place\_meeting(x,y,obj\_check)

{

if place\_meeting(x,y,obj\_start)

{

room\_goto(rm\_press)

}

}

//if they are not cheating and succesfully pass the finish line, it takes them to another room

else

{

if place\_meeting(x,y,obj\_start)

{

room\_goto(rm\_done)

}

}

//collision checking for hitting a wall

if place\_meeting(x,y,obj\_wall)

{

room\_goto(rm\_press)

}

//checks if the ebrake is not pressed

if !eb {

direction = image\_angle;

drDirL = 0;

drAngL = 0;

forSpdAcc = 2.88;

}

//to reset turning acceleration when not turning left

if !left {

turnAccL = 0;

}

//to reset turning acceleration when not turning right

if !right {

turnAccR = 0;

}

//brake deceleration start at zero when not braking, until you brake (mechanics are further down for that)

brakeDecel = 0;

////////////////////////////////////TURNING///////////////////////////////////////////////////////

//TURN RIGHT

if !left && !eb {

if right {

//WHILE REVERSING

//wanted to make the speed and acceleration at which you can turn different, depending on //your speed. It took me a very, very long time to tweak the numbers to the point where the //car flowed the way I wanted it to. The calculations are not simple but worked well.

//what I did for all these algorithms is I made different intervals of speed, and created a sort of //timer that would raise to a maximum value for (forward/reverse/turning etc.) acceleration, //and the timer would fall back down when you are no longer clicking the button.

//SPEED LESS THAN -0.50 & GREATER THAN OR EQUAL TO -1.00

if ((speed < -0.50) && (speed >= -1.00)) {

if (turnAccR < 0.4) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.022; //slowly climbs up each step. 60 steps in a second

image\_angle = (image\_angle + ((turnAccR \*0.3) + turnAccR)/1.2);

}

if turnAccR >= 0.4 { //if HAS reached max turning acceleration

turnAccR = 0.4; //stays at this value now (max)

image\_angle = (image\_angle + ((turnAccR \*0.3) + turnAccR)/1.2);

}

}

//SPEED LESS THAN -1.00 & GREATER THAN OR EQUAL TO -2.00

if ((speed < -1.00) && (speed >= -2.00)) {

if (turnAccR < 0.6) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.033;

image\_angle = (image\_angle + ((turnAccR \*0.45) + turnAccR)/1.2);

}

if turnAccR >= 0.6 { //if HAS reached max turning acceleration

turnAccR = 0.6;

image\_angle = (image\_angle + ((turnAccR \*0.45) + turnAccR)/1.2);

}

}

//SPEED LESS THAN -2.00 & GREATER THAN OR EQUAL TO -3.00

if ((speed < -2.00) && (speed >= -3.00)) {

if (turnAccR < 0.8) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.044;

image\_angle = (image\_angle + ((turnAccR \*0.6) + turnAccR)/1.2);

}

if turnAccR >= 0.8 { //if HAS reached max turning acceleration

turnAccR = 0.8;

image\_angle = (image\_angle + ((turnAccR \*0.6) + turnAccR)/1.2);

}

}

//WHILE GOING FORWARDS

//wanted to make the speed and acceleration at which you can turn different, depending on your speed.

//SPEED GREATER THAN 0.50 & LESS THAN OR EQUAL TO 1.00

if ((speed > 0.50) && (speed <= 1.00)) {

if (turnAccR < 0.4) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.022;

image\_angle = (image\_angle - ((turnAccR \*0.3) + turnAccR)/1.2);

}

if turnAccR >= 0.4 { //if HAS reached max turning acceleration

turnAccR = 0.4;

image\_angle = (image\_angle - ((turnAccR \*0.3) + turnAccR)/1.2);

}

}

//SPEED GREATER THAN 1.00 & LESS THAN OR EQUAL TO 2.00

if ((speed > 1.00) && (speed <= 2.00)) {

if (turnAccR < 0.6) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.033;

image\_angle = (image\_angle - ((turnAccR \*0.45) + turnAccR)/1.2);

}

if turnAccR >= 0.6 { //if HAS reached max turning acceleration

turnAccR = 0.6;

image\_angle = (image\_angle - ((turnAccR \*0.45) + turnAccR)/1.2);

}

}

//SPEED GREATER THAN 2.00 & LESS THAN OR EQUAL TO 3.00

if ((speed > 2.00) && (speed <= 3.00)) {

if (turnAccR < 0.8) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.044;

image\_angle = (image\_angle - ((turnAccR \*0.6) + turnAccR)/1.2);

}

if turnAccR >= 0.8 { //if HAS reached max turning acceleration

turnAccR = 0.8;

image\_angle = (image\_angle - ((turnAccR \*0.6) + turnAccR)/1.2);

}

}

//SPEED GREATER THAN 3.00 & LESS THAN OR EQUAL TO 4.00

if ((speed > 3.00) && (speed <= 4.00)) {

if (turnAccR < 1.1) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.061;

image\_angle = (image\_angle - ((turnAccR \*0.8) + turnAccR)/1.2);

}

if turnAccR >= 1.1 { //if HAS reached max turning acceleration

turnAccR = 1.1;

image\_angle = (image\_angle - ((turnAccR \*0.8) + turnAccR)/1.2);

}

}

//SPEED GREATER THAN 4.00 & LESS THAN OR EQUAL TO 5.00

if ((speed > 4.00) && (speed <= 5.00)) {

if (turnAccR < 1.4) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.078;

image\_angle = (image\_angle - ((turnAccR \*1.0) + turnAccR)/1.2);

}

if turnAccR >= 1.4 { //if HAS reached max turning acceleration

turnAccR = 1.4;

image\_angle = (image\_angle - ((turnAccR \*1.0) + turnAccR)/1.2);

}

}

//SPEED GREATER THAN 5.00

if (speed > 5.00) {

if (turnAccR < 1.8) { //if has NOT reached max turning acceleration

turnAccR = turnAccR + 0.1;

image\_angle = (image\_angle - ((turnAccR \*1.2) + turnAccR)/1.2);

}

if turnAccR >= 1.8 { //if HAS reached max turning acceleration

turnAccR = 1.8;

image\_angle = (image\_angle - ((turnAccR \*1.2) + turnAccR)/1.2);

}

}

}

}

//TURN LEFT

if !right && !eb {

if left {

//WHILE REVERSING

//SPEED LESS THAN -0.50 & GREATER THAN OR EQUAL TO -1.00

if ((speed < -0.50) && (speed >= -1.00)) {

if (turnAccL < 0.4) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.022;

image\_angle = (image\_angle - ((turnAccL \*0.3) + turnAccL)/1.2);

}

if turnAccL >= 0.4 { //if HAS reached max turning acceleration

turnAccL = 0.4;

image\_angle = (image\_angle - ((turnAccL \*0.3) + turnAccL)/1.2);

}

}

//SPEED LESS THAN -1.00 & GREATER THAN OR EQUAL TO -2.00

if ((speed < -1.00) && (speed >= -2.00)) {

if (turnAccL < 0.6) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.033;

image\_angle = (image\_angle - ((turnAccL \*0.45) + turnAccL)/1.2);

}

if turnAccL >= 0.6 { //if HAS reached max turning acceleration

turnAccL = 0.6;

image\_angle = (image\_angle - ((turnAccL \*0.45) + turnAccL)/1.2);

}

}

//SPEED LESS THAN -2.00 & GREATER THAN OR EQUAL TO -3.00

if ((speed < -2.00) && (speed >= -3.00)) {

if (turnAccL < 0.8) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.044;

image\_angle = (image\_angle - ((turnAccL \*0.6) + turnAccL)/1.2);

}

if turnAccL >= 0.8 { //if HAS reached max turning acceleration

turnAccL = 0.8;

image\_angle = (image\_angle - ((turnAccL \*0.6) + turnAccL)/1.2);

}

}

//WHILE GOING FORWARD

//SPEED GREATER THAN 0.50 & LESS THAN OR EQUAL TO 1.00

if ((speed > 0.50) && (speed <= 1.00)) {

if (turnAccL < 0.4) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.022;

image\_angle = (image\_angle + ((turnAccL \*0.3) + turnAccL)/1.2);

}

if turnAccL >= 0.4 { //if HAS reached max turning acceleration

turnAccL = 0.4;

image\_angle = (image\_angle + ((turnAccL \*0.3) + turnAccL)/1.2);

}

}

//SPEED GREATER THAN 1.00 & LESS THAN OR EQUAL TO 2.00

if ((speed > 1.00) && (speed <= 2.00)) {

if (turnAccL < 0.6) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.033;

image\_angle = (image\_angle + ((turnAccL \*0.45) + turnAccL)/1.2);

}

if turnAccL >= 0.6 { //if HAS reached max turning acceleration

turnAccL = 0.6;

image\_angle = (image\_angle + ((turnAccL \*0.45) + turnAccL)/1.2);

}

}

//SPEED GREATER THAN 2.00 & LESS THAN OR EQUAL TO 3.00

if ((speed > 2.00) && (speed <= 3.00)) {

if (turnAccL < 0.8) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.044;

image\_angle = (image\_angle + ((turnAccL \*0.6) + turnAccL)/1.2);

}

if turnAccL >= 0.8 { //if HAS reached max turning acceleration

turnAccL = 0.8;

image\_angle = (image\_angle + ((turnAccL \*0.6) + turnAccL)/1.2);

}

}

//SPEED GREATER THAN 3.00 & LESS THAN OR EQUAL TO 4.00

if ((speed > 3.00) && (speed <= 4.00)) {

if (turnAccL < 1.1) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.061;

image\_angle = (image\_angle + ((turnAccL \*0.8) + turnAccL)/1.2);

}

if turnAccL >= 1.1 { //if HAS reached max turning acceleration

turnAccL = 1.1;

image\_angle = (image\_angle + ((turnAccL \*0.8) + turnAccL)/1.2);

}

}

//SPEED GREATER THAN 4.00 & LESS THAN OR EQUAL TO 5.00

if ((speed > 4.00) && (speed <= 5.00)) {

if (turnAccL < 1.4) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.078;

image\_angle = (image\_angle + ((turnAccL \*1.0) + turnAccL)/1.2);

}

if turnAccL >= 1.4 { //if HAS reached max turning acceleration

turnAccL = 1.4;

image\_angle = (image\_angle + ((turnAccL \*1.0) + turnAccL)/1.2);

}

}

//SPEED GREATER THAN 5.00

if (speed > 5.00) {

if (turnAccL < 1.8) { //if has NOT reached max turning acceleration

turnAccL = turnAccL + 0.1;

image\_angle = (image\_angle + ((turnAccL \*1.2) + turnAccL)/1.2);

}

if turnAccL >= 1.8 { //if HAS reached max turning acceleration

turnAccL = 1.8;

image\_angle = (image\_angle + ((turnAccL \*1.2) + turnAccL)/1.2);

}

}

}

}

//////////////////////////FORWARD SPEED//////////////////////////////////////////////////

//IF DOWN & BRAKE & EBRAKE ARE NOT PRESSED, WE CAN MOVE FORWARDS BY CLICKING UP

if (!down && !brake && !eb) {

if up {

if (speed < 10) { //if speed is not at maximum value of 10 yet

speed = (((speed \* forSpdAcc) + forSpd) / 3); //you accelerate

}

if (speed > 10) { //if speed is at maximum value of 10

speed = 10; //speed remains 10 with no acceleration

}

}

}

//IF UP AND DOWN AND EBRAKE ARE NOT PRESSED, WE CAN SLOW DOWN BY PRESSING BRAKE

if (!up && !down && !eb) {

if brake {

if ((speed >= 0) && (speed < 0.30)) {

speed = 0;

}

//if moving FORWARD, slows us down until speed is at a value less than 0.30

if speed >= 0.30 {

brakeDecel = brakeDecel + 0.15;

speed = speed - brakeDecel;

}

if speed = 0 {

speed = 0;

}

//if moving BACKWARDS, slows us down until speed is at a value less than 0.30

if speed <= -0.30 {

brakeDecel = brakeDecel + 0.15;

speed = speed + brakeDecel;

}

if ((speed <= 0) && (speed > -0.30)) {

speed = 0;

}

}

}

//SLOW DOWN WHILE COASTING

if ((speed > 0.6) && !up) {

speed = speed - 0.025;

}

////////////////////////////BACKWARD SPEED////////////////////////////////////////////////////

//IF UP & BRAKE & EBRAKE ARE NOT PRESSED, WE CAN MOVE BACKWARD BY CLICKING UP

if (!up && !brake && !eb) {

if speed < 0.7 {

if down {

if (speed > -3) {

speed = ((speed \* backSpdAcc) + backSpd)/1.2;

}

if (speed < -3) {

speed = -3;

}

}

}

}

//SLOW DOWN WHILE COASTING

if ((speed < -0.6) && !down) {

speed = speed + 0.025;

}

/\*

I did not fully figure out the drifting mechanics. I was able to create code for the car to smoothly enter a drift, but could not quite figure out how to smoothly exit the drift. so I left this part out of the game

//FOR EBRAKE

//ENTER DRIFT MODE

if eb {

if left {

if speed > 7 {

speed = speed-0.11;

}

if (speed <= 2) {

drAngL = drAngL + 0.1;

drDirL = drDirL + 0.1;

image\_angle = (image\_angle + (5 - (drAngL \* 0.3)));

direction = (direction + (((drDirL \*1.15) + drDirL)/4.5));

}

if (speed > 2 && speed <= 3) {

drAngL = drAngL + 0.1;

drDirL = drDirL + 0.1;

image\_angle = (image\_angle + (5 - (drAngL \* 0.5)));

direction = (direction + (((drDirL \*1.15) + drDirL)/4));

}

if (speed > 3 && speed <= 4) {

drAngL = drAngL + 0.1;

drDirL = drDirL + 0.1;

image\_angle = (image\_angle + (5 - (drAngL \* 0.7)));

direction = (direction + (((drDirL \*1.15) + drDirL)/3.5));

}

if (speed > 4 && speed <=5) {

drAngL = drAngL + 0.1;

drDirL = drDirL + 0.1;

image\_angle = (image\_angle + (5 - (drAngL \* 0.9)));

direction = (direction + (((drDirL \*1.15) + drDirL)/3));

}

if (speed > 5) {

if (drAngL < 5) {

drAngL = drAngL + 0.1;

}

if (drDirL < 6) {

drDirL = drDirL + 0.1;

}

image\_angle = (image\_angle + (5 - (drAngL \* 1.1)));

direction = (direction + (((drDirL \*1.15) + drDirL)/2.5));

}

//if (speed <= 5) {

//image\_angle = (image\_angle + (5 - (4 \* 1.1)));

//direction = (direction + 4.3);

//}

}

//EXIT THE DRIFT SMOOTHLY

if keyboard\_check\_released(vk\_space) {

}

}

obj\_car\_Draw GUI\_1:

//draws the speed (multiplied by 10 to make realistic)

draw\_text(0,0,"Kmph: " + string(10\*speed));

obj\_void\_car:

(uses spr\_240sx, used for displaying the car without the algorithms being in effect)

obj\_void\_car\_Create\_1:

depth = -10 //for layer ordering

obj\_pavement:

(uses spr\_pavement)

obj\_pavement\_Create\_1:

depth = -7

obj\_start:

(uses spr\_start)

obj\_start\_Create\_1:

depth = -9

obj\_wall:

(uses spr\_wall)

obj\_wall\_Create\_1:

depth = -8

obj\_txt\_press:

(uses spr\_txt\_press)

obj\_txt\_press\_Step\_1:

score = 0 //starts your score at 0 before you press w to start driving

obj\_txt\_press\_press W-key\_1:

room\_goto(rm\_game) //takes you to room where you drive

obj\_score:

obj\_score\_Create\_1:

//sets the highscore to a large number until you load the score (incase you have never played //before and don’t have a highscore to load yet

var\_highScoreData = 1000000

scr\_loadHiScore() //loads the highscore

set Alarm 1 relative to 1

creates an alarm that ticks 60 times a second

obj\_score\_Alarm1\_1:

//adds 1 point every step, and there are 60 steps in a second. Later convert this into seconds by //dividing by 60

score = score + 1

set Alarm 1 relative to 1

//creates another alarm within the alarm

obj\_score\_Draw GUI\_1:

//saves highscore and loads highscore

scr\_SaveHiScore();

scr\_loadHiScore();

//set writing color to black

draw\_set\_color(c\_black)

//draws the score on the screen, divided by 60 to represent seconds/milliseconds

draw\_text(100,0,"Current Time: " + string(score/60));

{

draw\_set\_font(fnt\_segoe)

}

//draws the Highscore on the screen, divided by 60 to represent seconds/milliseconds

draw\_text(100,30,"Best Time: " + string(var\_highScoreData/60));

{

draw\_set\_font(fnt\_segoe)

}

obj\_check:

(uses spr\_check)

blank, needed to create the object to implement into the room and into other objects

obj\_txt\_press2:

(uses spr\_txt\_press2)

obj\_txt\_press2\_Create\_1:

//used to display the prompt to press E, and also to save your new score after completing a //successful race to the HIGHSCORE if it is lower than the highscore.

var\_highScoreData = score

scr\_loadHiScore()

obj\_txt\_press2\_Draw GUI\_1:

//saves highscore and loads highscore

scr\_SaveHiScore();

scr\_loadHiScore();

//set writing color to black

draw\_set\_color(c\_black)

//draws the score on the screen, divided by 60 to represent seconds/milliseconds

draw\_text(100,0,"Current Time: " + string(score/60));

{

draw\_set\_font(fnt\_segoe)

}

//draws the Highscore on the screen, divided by 60 to represent seconds/milliseconds

draw\_text(290,30,"Best Time: " + string(var\_highScoreData/60));

{

draw\_set\_font(fnt\_segoe)

}

obj\_txt\_press2\_press E-key\_1:

//go to the room that prompts you to press W to start racing

room\_goto(rm\_press)

obj\_logo:

(uses spr\_logo)

blank, needed for displaying the logo in rooms

ROOMS

rm\_press urges you to press W to start the race (go to rm\_game), and displays my logo.

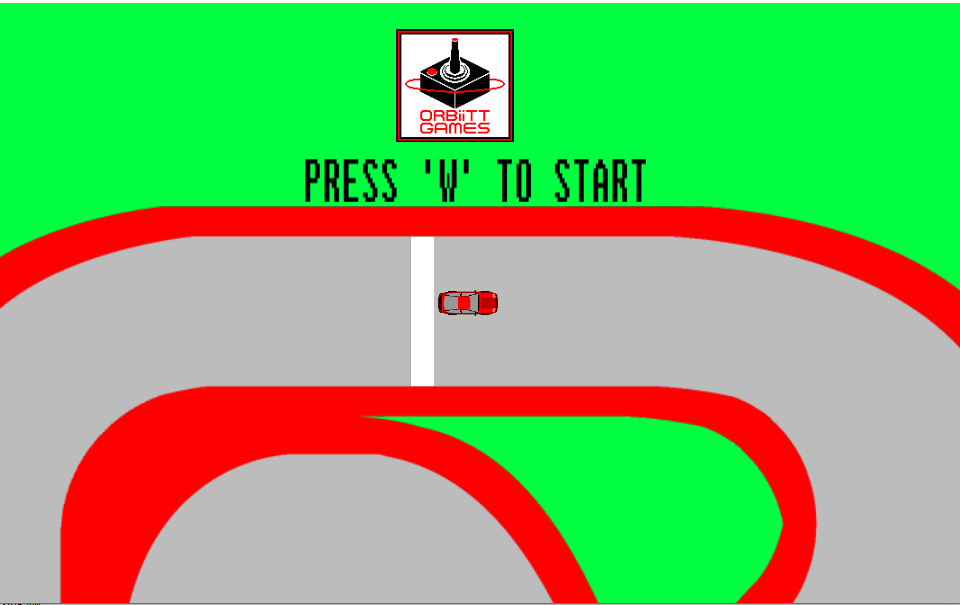
rm\_game is where you drive the car. If you hit a wall you return to rm\_press to retry, and if you

successfully complete the race, you go to rm\_done. This room also has a follow camera set up, that keeps your car in the centre of the screen as you race.

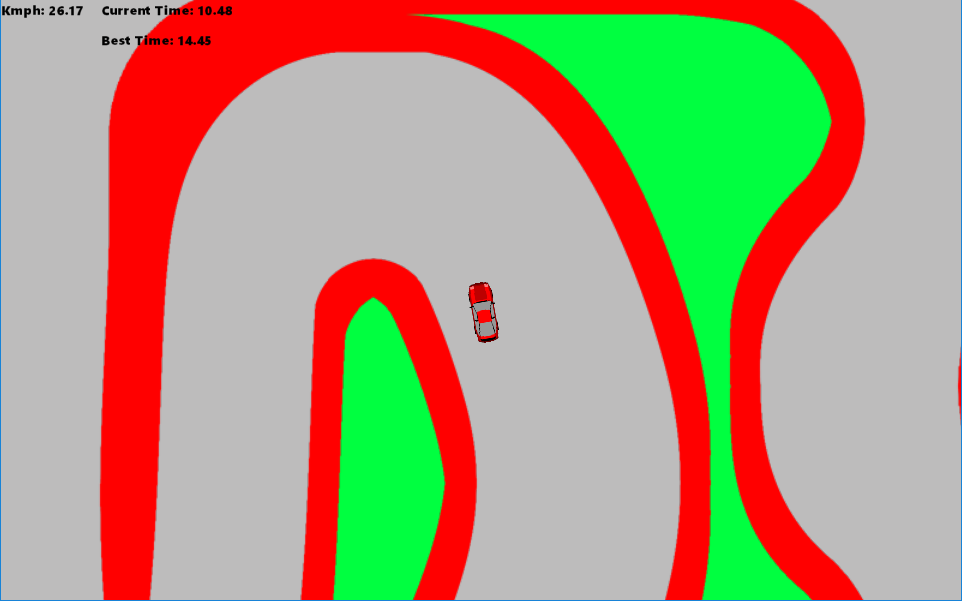
rm\_done urges you to press E (to go to rm\_press), and displays my logo.

Screenshots of Rooms:

rm\_press:



rm\_game:



rm\_done:



Conclusion:

This was by far my favourite project I’ve been assigned for homework since starting my studies of computer science at TWU. I really enjoyed working on this game, and because I found it so fun I was able to put a large amount of time and work into this project. It really tested my abilities to the point where I was stumped on certain things for hours on end (such as saving and loading Highscores that carry on even if you close and re-open the game, which I eventually figured out). Because of this project I now know that I would love to work in the gaming industry after graduation. It was great to be able to implement my love for art by designing the sprite for the car and other sprites, and the logo “ORBiiTT GAMES.” I also loved being able to implement my passion for gaming into this project. If I had more time, I would have loved to make more pixel art, to make the game look much better with a dynamic crowd watching you race and a more realistic looking race track and grass. I also wish I was able to create sound effects and music, as I have been producing music for over 6 years now on my computer which would add a nice touch to the game. I would have also loved to finish the drift mechanics, add new tracks, unlockable cars with different attributes and appearance. Possibly create smoke animation and skid marks that follow behind the car on sharp turns. I also believe I could create AI to race against, which would add a very fun dynamic to the game. Maybe create a different mode, such as a drag-racing mode with timing-based gear shifting. Sadly, I did not have enough time to implement all these things, but I may continue this project on my own time, if I find time.

This game is downloadable as a .exe at this link:

<http://www.mediafire.com/file/yj194eb6ez799y5/Time_Trial_Racing_Game_-_Drift.exe/file>