using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Audio;

using Microsoft.Xna.Framework.Content;

using Microsoft.Xna.Framework.GamerServices;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

using Microsoft.Xna.Framework.Media;

using System.IO;

namespace Space\_Shooter

{

/// <summary>

/// This is the main type for your game

/// </summary>

public class Game1 : Microsoft.Xna.Framework.Game

{

int enemyCount = 4;

int speed;

float laserFireStartTime = 0.0f;

int waveIndex;

TimeSpan totalTimeSurvived;

TimeSpan yourBest;

StreamReader readBestTime;

StreamWriter writeBestTime;

GraphicsDeviceManager graphics;

SpriteBatch spriteBatch;

Texture2D space;

Texture2D ship;

Texture2D laserBeam;

Texture2D backgroundPlanet1;

Texture2D backgroundPlanet2;

Texture2D backgroundPlanet3;

Texture2D[] enemyShip1;

Texture2D[] enemyMissiles;

SpriteFont reloadingT;

SpriteFont scoreT;

SpriteFont lostT;

SoundEffect playerFireSound;

SoundEffectInstance playerFireSoundIns;

Song backgroundMusic;

Vector2 laserPos = Vector2.Zero;

Vector2 missilePos;

Vector2[] enemyPos;

Rectangle[] enemyBounds;

Rectangle laserBounds;

Rectangle playerBounds;

Rectangle[] enemyMissileBounds;

bool moveUp = true;

bool moveDown = true;

bool moveLeft = true;

bool moveRight = true;

bool fire = false;

bool reloaded = true;

bool showText = false;

bool lost = false;

bool enemyShoot = false;

bool enemyReloaded = true;

bool[] enemyKilled;

bool[] missileToShoot = new bool[4];

bool allKilled = false;

bool recordTime = true;

bool muted = false;

bool paused = false;

string message;

string debugMessage = "Debug";

public Game1()

{

graphics = new GraphicsDeviceManager(this);

Content.RootDirectory = "Content";

}

protected override void Initialize()

{

graphics.PreferredBackBufferWidth = 800;

graphics.PreferredBackBufferHeight = 750;

graphics.ApplyChanges();

enemyShip1 = new Texture2D[enemyCount];

enemyPos = new Vector2[enemyCount];

enemyKilled = new bool[enemyCount];

enemyBounds = new Rectangle[enemyCount];

enemyMissiles = new Texture2D[enemyCount];

enemyMissileBounds = new Rectangle[enemyCount];

this.Window.Title = "Universal War 1";

waveIndex = 1;

speed = 1;

message = "Reloading Laser Gun";

base.Initialize();

}

protected override void LoadContent()

{

float currentPosX = 0; ;

float currentPosY = 0;

// Create a new SpriteBatch, which can be used to draw textures.

spriteBatch = new SpriteBatch(GraphicsDevice);

space = Content.Load<Texture2D>("Space");

ship = Content.Load<Texture2D>("Spaceship");

laserBeam = Content.Load<Texture2D>("Laserbeam");

reloadingT = Content.Load<SpriteFont>("Reloading");

scoreT = Content.Load<SpriteFont>("Score");

lostT = Content.Load<SpriteFont>("Lost");

backgroundPlanet1 = Content.Load<Texture2D>("BackgroundPlanet1");

backgroundPlanet2 = Content.Load<Texture2D>("BackgroundPlanet2");

backgroundPlanet3 = Content.Load<Texture2D>("BackgroundPlanet3");

playerFireSound = Content.Load<SoundEffect>("RayGun");

backgroundMusic = Content.Load<Song>("Background Music");

playerFireSoundIns = playerFireSound.CreateInstance();

for (int a = 0; a < enemyShip1.Length; a++)

{

enemyShip1[a] = Content.Load<Texture2D>("Enemy ship 1");

}

for (int b = 0; b < enemyMissiles.Length; b++)

{

enemyMissiles[b] = Content.Load<Texture2D>("Missile");

}

for (int c = 0; c < enemyMissileBounds.Length; c++)

{

enemyMissileBounds[c] = new Rectangle((int)enemyBounds[c].X, enemyBounds[c].Y, laserBeam.Width + 1, laserBeam.Height + 1);

}

laserBounds = new Rectangle(0, 0, laserBeam.Width, laserBeam.Height);

playerBounds = new Rectangle((int)graphics.GraphicsDevice.Viewport.Width / 2, (int)graphics.GraphicsDevice.Viewport.Height - 150, 50, 75);

//...............Positioning Enemies...................

for (int b = 0; b < enemyPos.Length; b++)

{

if (b % 2 == 0)

{

currentPosY = 0;

}

else if (b % 2 == 1)

{

currentPosY = enemyShip1[b].Height;

}

enemyPos[b] = new Vector2(currentPosX + enemyShip1[b].Width, currentPosY);

currentPosX = enemyPos[b].X;

}

for (int a = 0; a < enemyBounds.Length; a++)

{

enemyBounds[a] = new Rectangle((int)enemyPos[a].X, (int)enemyPos[a].Y, 50, 75);

}

//...................Playing Background Music.............

MediaPlayer.Play(backgroundMusic);

}

protected override void UnloadContent()

{

// TODO: Unload any non ContentManager content here

}

protected override void Update(GameTime gameTime)

{

//.............Pausing the game...................

if (Keyboard.GetState().IsKeyDown(Keys.P) && !paused)

{

paused = true;

}

else if (Keyboard.GetState().IsKeyDown(Keys.P) && paused)

{

paused = false;

}

if (this.IsActive && !paused)

{

//..................Muting the sound................

if (Keyboard.GetState().IsKeyDown(Keys.M) && !muted)

{

MediaPlayer.Stop();

muted = true;

}

//..............UnMuting The Sound...............

if (Keyboard.GetState().IsKeyDown(Keys.N) && muted)

{

MediaPlayer.Play(backgroundMusic);

muted = false;

}

//..................Keeping Track Of Total Time Survived..........

if (recordTime)

{

totalTimeSurvived = gameTime.TotalGameTime;

}

//...............Ship Movement Control.................

if (Keyboard.GetState().IsKeyDown(Keys.Escape))

{

this.Exit();

}

if (Keyboard.GetState().IsKeyDown(Keys.Left) && moveLeft)

{

playerBounds.X -= 6;

}

if (Keyboard.GetState().IsKeyDown(Keys.Right) && moveRight)

{

playerBounds.X += 6;

}

if (Keyboard.GetState().IsKeyDown(Keys.Up) && moveUp)

{

playerBounds.Y -= 4;

}

if (Keyboard.GetState().IsKeyDown(Keys.Down) && moveDown)

{

playerBounds.Y += 4;

}

//...............Keeping the ship in the window................

if (playerBounds.X <= 0)

{

moveLeft = false;

}

else

{

moveLeft = true;

}

if (playerBounds.X >= graphics.GraphicsDevice.Viewport.Width - (ship.Width / 2))

{

moveRight = false;

}

else

{

moveRight = true;

}

if (playerBounds.Y <= 0)

{

moveUp = false;

}

else

{

moveUp = true;

}

if (playerBounds.Y >= (graphics.GraphicsDevice.Viewport.Height - 100))

{

moveDown = false;

}

else

{

moveDown = true;

}

//.............Ship firing mechanism............

if (Keyboard.GetState().IsKeyDown(Keys.Space) && reloaded && !lost)

{

fire = true;

reloaded = false;

if (!muted)

{

playerFireSoundIns.Play();

}

//.............Laser positioning.............

laserBounds.X = (int)playerBounds.X;

laserBounds.Y = (int)playerBounds.Y - laserBeam.Height;

//............Shows Text..................

if (!reloaded)

{

showText = true;

}

}

//..............Laser Motion.................

if (fire)

{

laserBounds.Y -= 10;

}

//..............Checks if laser beam has been destroyed.............

if (laserBounds.Y <= 0)

{

reloaded = true;

showText = false;

laserFireStartTime = 0;

}

else

{

reloaded = false;

}

//...............Moving enemy Ships................

for (int b = 0; b < enemyPos.Length; b++)

{

enemyBounds[b].Y += speed;

}

//..............Checking If Going out of Window.............

for (int a = 0; a < enemyPos.Length; a++)

{

if (enemyPos[a].Y == graphics.GraphicsDevice.Viewport.Height - 85)

{

lost = true;

}

}

//.................Checking if enemy has been hit............

for (int a = 0; a < enemyBounds.Length; a++)

{

if (enemyBounds[a].Intersects(laserBounds))

{

enemyKilled[a] = true;

enemyBounds[a] = default(Rectangle);

}

}

//.................Making The Enemies Shoot...............

for (int a = 0; a < enemyShip1.Length; a++)

{

if (playerBounds.Y > enemyBounds[a].Y && playerBounds.X <= enemyBounds[a].X + enemyShip1[0].Width && playerBounds.X >= enemyBounds[a].X && enemyReloaded && enemyKilled[a] != true)

{

missileToShoot[a] = true;

enemyShoot = true;

enemyMissileBounds[a].X = enemyBounds[a].X;

enemyMissileBounds[a].Y = enemyBounds[a].Y;

enemyReloaded = false;

}

}

//...................Propelling Enemy Missile.................

for (int b = 0; b < missileToShoot.Length; b++)

{

if (missileToShoot[b])

{

enemyMissileBounds[b].Y += 6;

}

}

//............Checking If Enemy Missile Is Out Of The Window...........

for (int c = 0; c < missileToShoot.Length; c++)

{

if (missileToShoot[c])

{

if (enemyMissileBounds[c].Y >= graphics.GraphicsDevice.Viewport.Height)

{

enemyReloaded = true;

}

}

}

//..................Checking If Player Has Been Hit................

for (int a = 0; a < enemyMissileBounds.Length; a++)

{

if (enemyMissileBounds[a].Intersects(playerBounds))

{

lost = true;

enemyReloaded = true;

recordTime = false;

MediaPlayer.Stop();

}

}

for (int b = 0; b < enemyBounds.Length; b++)

{

if (enemyBounds[b].Intersects(playerBounds))

{

lost = true;

recordTime = false;

MediaPlayer.Stop();

}

}

//...................Checking If Enemies Have Reached The Base...............

for (int a = 0; a < enemyBounds.Length; a++)

{

if (enemyBounds[a].Y >= (graphics.GraphicsDevice.Viewport.Height - enemyShip1[0].Height) + enemyShip1[0].Height)

{

lost = true;

recordTime = false;

MediaPlayer.Stop();

NextEnemyWave();

}

}

//...............Checking If All Enemies Have Died...........

for (int a = 0; a < enemyKilled.Length; a++)

{

allKilled = true;

if (!enemyKilled[a] && a != (enemyKilled.Length - 1))

{

allKilled = false;

}

else if (a == (enemyKilled.Length - 1) && !enemyKilled[a])

{

allKilled = false;

}

else if (a == (enemyKilled.Length - 1) && enemyKilled[a] & (!enemyKilled[0] || !enemyKilled[1] || !enemyKilled[2]))

{

allKilled = false;

}

}

if (allKilled)

{

NextEnemyWave();

}

if (lost)

{

BestTime();

}

}

base.Update(gameTime);

}

protected override void Draw(GameTime gameTime)

{

if (!lost)

{

spriteBatch.Begin(SpriteSortMode.Deferred, BlendState.AlphaBlend);

//......................Rendering Space Background...................

spriteBatch.Draw(space, new Rectangle(0, 0, graphics.GraphicsDevice.Viewport.Width, graphics.GraphicsDevice.Viewport.Height), Color.White);

//..............Rendering Background Sprites..................

spriteBatch.Draw(backgroundPlanet1, new Rectangle(200, graphics.GraphicsDevice.Viewport.Height / 2 - backgroundPlanet1.Height, 80, 80), Color.White);

spriteBatch.Draw(backgroundPlanet2, new Rectangle(graphics.GraphicsDevice.Viewport.Width - 200, 50, 200, 200), Color.YellowGreen);

spriteBatch.Draw(backgroundPlanet3, new Rectangle(100, graphics.GraphicsDevice.Viewport.Height - 125, 125, 125), Color.Violet);

spriteBatch.Draw(backgroundPlanet1, new Rectangle(graphics.GraphicsDevice.Viewport.Width - 220, graphics.GraphicsDevice.Viewport.Height - 220, 220, 220), Color.SandyBrown);

//...............Drawing Main Game Objects....................

// spriteBatch.DrawString(reloadingT, debugMessage, new Vector2(graphics.GraphicsDevice.Viewport.Width / 2, 0), Color.BlanchedAlmond);

spriteBatch.Draw(ship, playerBounds, Color.White);

if (fire)

{

spriteBatch.Draw(laserBeam, laserBounds, Color.White);

}

for (int a = 0; a < enemyShip1.Length; a++)

{

if (enemyKilled[a] == true)

{

}

else

{

spriteBatch.Draw(enemyShip1[a], enemyBounds[a], Color.White);

}

}

if (enemyShoot)

{

for (int a = 0; a < missileToShoot.Length; a++)

{

if (missileToShoot[a])

{

spriteBatch.Draw(enemyMissiles[a], enemyMissileBounds[a], Color.White);

}

}

}

if (showText)

{

spriteBatch.DrawString(reloadingT, message, new Vector2(graphics.GraphicsDevice.Viewport.Width - 200, graphics.GraphicsDevice.Viewport.Height - 100), Color.Red);

}

spriteBatch.DrawString(scoreT, totalTimeSurvived.ToString(), new Vector2(graphics.GraphicsDevice.Viewport.Width / 2 - 20, 0), Color.Yellow);

spriteBatch.End();

}

else

{

graphics.GraphicsDevice.Clear(Color.Red);

spriteBatch.Begin();

spriteBatch.DrawString(lostT, "YOU LOST", new Vector2((graphics.GraphicsDevice.Viewport.Width / 2) - 150, graphics.GraphicsDevice.Viewport.Height / 2 - 50), Color.Black);

spriteBatch.DrawString(scoreT, "TIME SURVIVED: " + totalTimeSurvived.ToString(), new Vector2((graphics.GraphicsDevice.Viewport.Width / 2) - 150, graphics.GraphicsDevice.Viewport.Height / 2 + 100), Color.Gold);

spriteBatch.DrawString(scoreT, "YOUR BEST TIME: " + yourBest.ToString(), new Vector2((graphics.GraphicsDevice.Viewport.Width / 2) - 150, graphics.GraphicsDevice.Viewport.Height / 2 + 150), Color.Gold);

spriteBatch.DrawString(scoreT, "A game developed by Aryan", new Vector2((graphics.GraphicsDevice.Viewport.Width / 2) - 110, graphics.GraphicsDevice.Viewport.Height - 50), Color.Moccasin);

spriteBatch.End();

}

//......................Finished Drawing.........................

base.Draw(gameTime);

}

void NextEnemyWave()

{

waveIndex++;

if(waveIndex <= 5 && speed < 5)

{

speed++;

}

else if(waveIndex <= 10 && waveIndex > 5 && speed < 5)

{

speed++;

}

else if(waveIndex <= 15 && waveIndex > 10 && speed < 5)

{

speed++;

}

else if(waveIndex <= 20 && waveIndex > 15 && speed < 5)

{

speed++;

}

int currentX = 0;

int currentY = 0;

for (int a = 0; a < enemyBounds.Length; a++)

{

if (a % 2 == 0)

{

currentY = 0;

}

else

{

currentY = enemyShip1[a].Height;

}

enemyBounds[a] = new Rectangle((int)currentX + enemyShip1[0].Width, currentY, 50, 75);

currentX = enemyBounds[a].X;

}

for (int b = 0; b < enemyKilled.Length; b++)

{

enemyKilled[b] = false;

}

}

void BestTime()

{

string firstFolderPath = @"Gameologist Games";

string secondFolderPath = @"Gameologist Games\\Universal War 1";

if (!Directory.Exists(firstFolderPath))

{

Directory.CreateDirectory(firstFolderPath);

}

if (!Directory.Exists(secondFolderPath))

{

Directory.CreateDirectory(secondFolderPath);

writeBestTime = new StreamWriter("Gameologist Games\\Universal War 1\\HighScore");

writeBestTime.WriteLine(TimeSpan.Zero);

writeBestTime.Close();

}

readBestTime = new StreamReader("Gameologist Games\\Universal War 1\\HighScore");

yourBest = TimeSpan.Parse(readBestTime.ReadLine());

readBestTime.Close();

if(totalTimeSurvived > yourBest)

{

writeBestTime = new StreamWriter("Gameologist Games\\Universal War 1\\HighScore");

writeBestTime.WriteLine(totalTimeSurvived);

writeBestTime.Close();

}

}

}

}