```
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;

namespace Frogger
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Texture2D frogger, black, purple;
        Rectangle rect_frogger, rect_black, rect_purple;
        Texture2D[] cars;
        Rectangle[] rect_cars;
        String[] cardirection;
        int[] carspeed;
        int speedofcars = 4;
        int froggerspeed = 1;
        int numberofcars = 10;
        int lives = 3;
        SpriteFont sf;


        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting to run.
        /// This is where it can query for any required services and load any non-graphic
        /// related content.  Calling base.Initialize will enumerate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here

            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            // TODO: use this.Content to load your game content here
            sf = Content.Load<SpriteFont>("sf");
            frogger = Content.Load<Texture2D>("frogger");
            rect_frogger.Width = frogger.Width;
            rect_frogger.Height = frogger.Height;
```

```
            rect_frogger.X = graphics.GraphicsDevice.Viewport.Width / 2;
            rect_frogger.Y = graphics.GraphicsDevice.Viewport.Height - frogger.Height;
            black = Content.Load<Texture2D>("black");
            rect_black.Width = graphics.GraphicsDevice.Viewport.Width;
            rect_black.Height = 159;
            rect_black.X = 0;
            rect_black.Y = 300;
            purple = Content.Load<Texture2D>("purple");
            rect_purple.Width = graphics.GraphicsDevice.Viewport.Width;
            rect_purple.Height = purple.Height;
            rect_purple.X = 0;
            rect_purple.Y = 459;

            cars = new Texture2D[numberofcars];
            rect_cars = new Rectangle[numberofcars];
            cardirection = new String[numberofcars];
            carspeed = new int[numberofcars];
            // loading 0 to 3
            for (int counter = 0; counter <= 3; counter = counter + 1)
            {
                cars[counter] = Content.Load<Texture2D>("car1");
                rect_cars[counter].Width = cars[counter].Width;
                rect_cars[counter].Height = cars[counter].Height;
                rect_cars[counter].Y = 420;
                rect_cars[counter].X = 100 * counter;
                cardirection[counter] = "R";
                carspeed[counter] = 4;
            }
            // loading 4 to 6
            for (int counter = 4; counter <= 6; counter = counter + 1)
            {
                cars[counter] = Content.Load<Texture2D>("car2");
                rect_cars[counter].Width = cars[counter].Width;
                rect_cars[counter].Height = cars[counter].Height;
                rect_cars[counter].Y = 390;
                rect_cars[counter].X = 200 + 100 * counter;
                cardirection[counter] = "L";
                carspeed[counter] = 4;
            }
            // loading 7 to 9
            for (int counter = 7; counter <= 9; counter = counter + 1)
            {
                cars[counter] = Content.Load<Texture2D>("car3");
                rect_cars[counter].Width = cars[counter].Width;
                rect_cars[counter].Height = cars[counter].Height;
                rect_cars[counter].Y = 360;
                rect_cars[counter].X = 100 * (counter - 6);
                cardirection[counter] = "L";
                carspeed[counter] = 5;
            }


        }

        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// all content.
        /// </summary>
        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
```

```
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here
            KeyboardState kb = Keyboard.GetState();
            if (kb.IsKeyDown(Keys.Up))
                rect_frogger.Y = rect_frogger.Y - froggerspeed;
            if (kb.IsKeyDown(Keys.Down))
                rect_frogger.Y = rect_frogger.Y + froggerspeed;
            if (kb.IsKeyDown(Keys.Left))
                rect_frogger.X = rect_frogger.X - froggerspeed;
            if (kb.IsKeyDown(Keys.Right))
                rect_frogger.X = rect_frogger.X + froggerspeed;

            // moving my cars
            for (int counter = 0; counter <= numberofcars - 1; counter++)
            {
                if (cardirection[counter].Equals("R"))
                    rect_cars[counter].X = rect_cars[counter].X + carspeed[counter];
                else
                    rect_cars[counter].X = rect_cars[counter].X - carspeed[counter];
            }
            // checking to see if they are off the screen
            for (int counter = 0; counter <= numberofcars - 1; counter++)
            {
                if (rect_cars[counter].X > graphics.GraphicsDevice.Viewport.Width)
                    rect_cars[counter].X = 0;
                if (rect_cars[counter].X < 0)
                    rect_cars[counter].X = graphics.GraphicsDevice.Viewport.Width - rect_cars[counter].
    Width;

            }
            // check to see if cars have hit frogger
            for (int counter = 0; counter <= numberofcars - 1; counter++)
            {
                if (rect_cars[counter].Intersects(rect_frogger))
                {
                    rect_frogger.X = graphics.GraphicsDevice.Viewport.Width / 2;
                    rect_frogger.Y = graphics.GraphicsDevice.Viewport.Height - frogger.Height;
                    lives = lives - 1;
                    if (lives == 0)
                        this.Exit();
                }
            }
            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // TODO: Add your drawing code here
            spriteBatch.Begin();
            spriteBatch.Draw(black, rect_black, Color.White);
            spriteBatch.Draw(purple, rect_purple, Color.White);
            spriteBatch.Draw(frogger, rect_frogger, Color.White);
            for (int counter = 0; counter <= numberofcars - 1; counter = counter + 1)
```

```
            {
                spriteBatch.Draw(cars[counter], rect_cars[counter], Color.White);
            }
            spriteBatch.DrawString(sf, "Lives: " + lives.ToString(), new Vector2(0, 0), Color.Red);
            spriteBatch.End();

            base.Draw(gameTime);
        }
    }
}
```