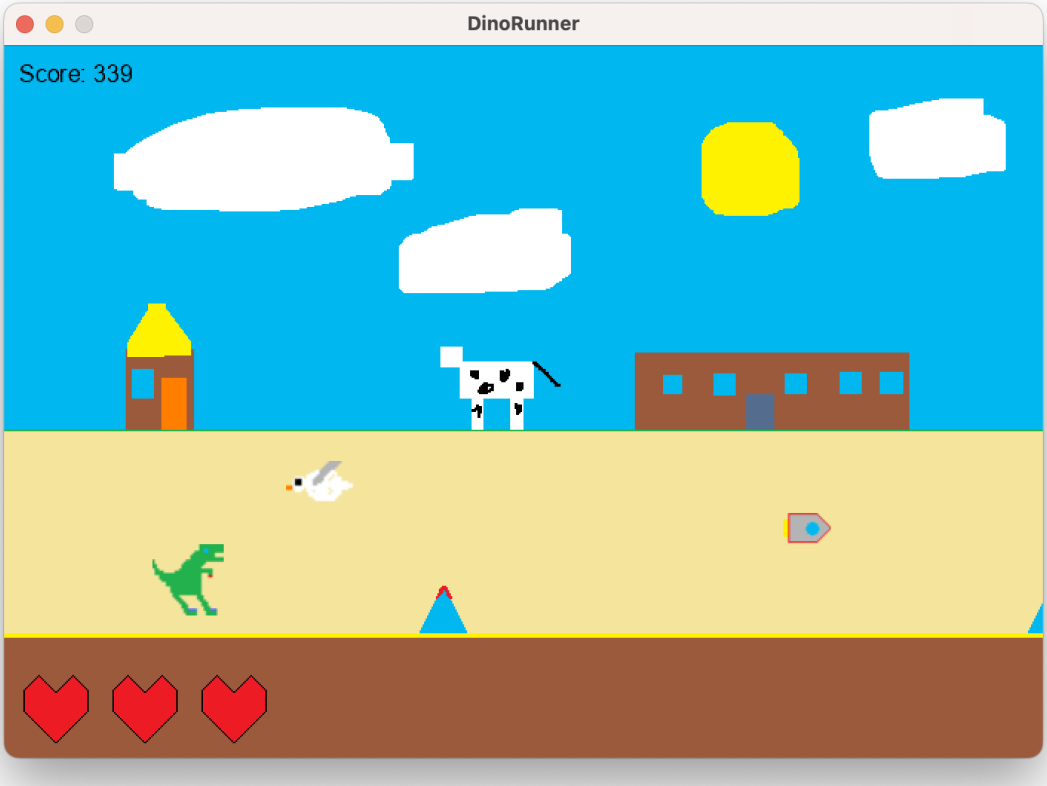
User Documentation

Dino Runner Game

Dino Runner is a simple 2D side-scrolling game where the player controls a dinosaur character to avoid obstacles and birds. The player can make the dinosaur jump and shoot rockets to eliminate flying birds.



How to Play

1. Press Enter to start the game.
2. Press Space to make the dinosaur jump.
3. Press F to shoot rockets.
4. Avoid colliding with obstacles and birds.
5. The game ends when the dinosaur loses all its health or collides with an obstacle/bird. Press R to restart the game.

Scoring

The player's score increases as they progress in the game. The game speed also increases with the score.

Technical Documentation

This section provides an overview of the game classes and their main responsibilities.

`Program.cs`

This class contains the main entry point of the game. It initializes and runs the game.

`Game1.cs`

This class is the main game class and is responsible for managing game states, loading content, updating game logic, and drawing game objects.

Properties and Fields

- **_graphics**: Graphics device manager.
- **_spriteBatch**: Sprite batch used to draw textures.
- **_gameSpeed**: The speed of the game.
- **_gameScore**: The score of the game.
- **_player**: The player object.
- **_ground**: The ground object.
- **_obstacles**: List of obstacle objects.
- **_birds**: List of bird objects.
- **_obstacleSpawnTimer**: Timer to control obstacle spawning.
- **_obstacleSpawnInterval**: Interval between obstacle spawns.
- **_birdSpawnTimer**: Timer to control bird spawning.
- **_birdSpawnInterval**: Interval between bird spawns.

Methods

- **Initialize**(): Initializes the game state, obstacles, and birds.
- **LoadContent**(): Loads game content such as textures, fonts, etc.
- **UpdateObstacles**(GameTime): Updates obstacles and checks for collisions with the player.
- **UpdateBirds**(GameTime): Updates birds and checks for collisions with the player.
- **CheckRocketBirdCollisions**(): Checks for collisions between rockets and birds.
- **Update**(GameTime): Updates game logic based on the current game state.
- **Draw**(GameTime): Draws game objects and UI elements.

`Player.cs`

This class represents the player's dinosaur character.

Properties and Fields

- **_playerState**: The current state of the player (waiting, running, jumping, or dead).
- **_waitingTexture**, **_runningTexture1**, **_runningTexture2**, **_deadTexture**: Textures for different player states.
- **_currentTexture**: The texture currently being displayed.
- **_position**: The position of the player on the screen.
- **_velocity**: The player's velocity.
- **_rocketTexture**: The texture for the rocket.
- **Rockets**: List of rockets fired by the player.
- **Health**: The player's health.
- **_heartTexture**: The texture for the player's health icon.

Methods

- **Collide**(): Handles player collision with obstacles or birds.
- **Update**(GameTime): Updates the player's state and position.
- **Shoot**(GameTime): Shoots a rocket if the player is not dead and the cooldown has expired.
- **Draw**(SpriteBatch): Draws the player and its rockets.

`Rocket.cs`

The **Rocket** class represents a rocket object in the game. It has a position, a texture representing the rocket image, and a speed.

Properties

- **Position**: A **Vector2** representing the rocket's position on the screen.
- **Bounds**: A **Rectangle** representing the rocket's bounding box, used for collision detection.

Constructor

- **Rocket**(**Texture2D** texture, **Vector2** position): Creates a new **Rocket** object with the provided texture and position.

Methods

- **Update**(GameTime gameTime): **Updates** the rocket's position based on its speed and the elapsed game time.
- **Draw**(SpriteBatch spriteBatch): **Draws** the rocket on the screen using the provided sprite batch.

`Bird.cs`

The **Bird** class represents a bird object in the game. It has position, two textures for animating the bird's flight, and an animation timer.

Properties

- **Bounds**: A **Rectangle** representing the bird's bounding box, used for collision detection.
- **Position**: A **Vector2** representing the bird's position on the screen.
- **Width**: A **float** representing the width of the bird texture.

Constructor

Bird(**ContentManager** content, **Vector2** position): Creates a new **Bird** object with the provided content manager to load textures and the initial position.

Methods

- **Update**(**GameTime** gameTime, **float** speed): **Updates** the bird's position based on the provided speed and updates the animation timer based on the elapsed game time.
- **Draw**(**SpriteBatch** spriteBatch): **Draws** the bird on the screen using the provided sprite batch, animating between two textures.
- **UpdateAnimation**(**GameTime** gameTime): **Updates** the bird's animation, toggling between two textures based on the animation timer.

`Obstacle.cs`

The **Obstacle** class represents an obstacle object in the game. It has a position and a texture representing the obstacle image.

Properties

- **Bounds**: A **Rectangle** representing the obstacle's bounding box, used for collision detection.
- **Position**: A **Vector2** representing the obstacle's position on the screen.
- **Width**: A **float** representing the width of the obstacle texture.

Constructor

- **Obstacle**(**ContentManager** content, **Vector2** position, **int** obstacleType): Creates a new Obstacle object with the provided content manager to load the texture, initial position, and obstacle type.
-

Methods

- **Update**(**float** speed): **Updates** the obstacle's position based on the provided speed.
- **Draw**(**SpriteBatch** spriteBatch): **Draws** the obstacle on the screen using the provided sprite batch.

`Ground.cs`

The **Ground** class represents the ground in the game. It has a texture representing the ground image and a list of positions for ground tiles.

Constructor

- **Ground**(**ContentManager** content): **Creates** a new **Ground** object with the provided content manager to load the texture.

Methods

- **Draw**(**SpriteBatch** spriteBatch): **Draws** the ground on the screen using the provided sprite batch, with multiple ground tiles arranged horizontally.