Developer Introduction to Farm Equipment Simulator 2021

Introduction

Farm Equipment Simulator 2021 (FES) is a single-player arcade style game written in MATLAB. In it, the user flies a rocket across a crop field, plowing the ground with the exhaust. They can hit cows along the way to gain more fuel to continue plowing. The game ends when the rocket crashes.

The SpriteKit Framework

FES uses the SpriteKit Framework extensively. This makes displaying images to the screen a lot easier and faster than using MATLAB's built-in imrotate() and imtranslate() functions, which caused strange artifacts and scaling in early builds of the game.

As such, here's a quick introduction to the SpriteKit Framework:

The SpriteKit Framework is used here to display most of the game on screen. We first create a Game object that will handle displaying our Sprites and Background and creating a MATLAB Figure. We then create a Background object with a .png image. It can be scrolled in four directions and will automatically tile the image to fill the Figure window. We then create Sprite objects by passing in a .png image. Sprites are used to represent the rocket, the cows, and the title/pause/crash screens.

The Sprite objects have a few built-in properties:
-Location: Location is a (x,y) vector in pixels that represents where the image should be drawn on screen. If we change this in action(), SpriteKit takes care of moving the Sprite.
-Angle: Angle is a scalar value in degrees that represents the rotation of the Sprite. SpriteKit will take care of displaying the image properly if we change this value.
-State: Different States can show different images. This is used to enable and disable the cows (they have a state where they are invisible, and we check state when they collide with the rocket - if they are invisible, we do nothing.)

We also add our own custom properties using the addprop() function. This allows us to store any value we want as a property of the Sprite object. Using this method, the rocket can remember its fuel mass, gravity, thrust, and more. Virtually all values used for physics are stored to the Sprite this way.

Main Game Loop

The main game follows a simple loop when playing:

1) Read the input buffer
2) Pause the game if needed
3) Apply rocket physics
4) Draw gauges, altimeter, and score on screen
5) Update exhaust particles
6) Scroll the background
7) Update the cow / tractor (they're one thing.)

Game Functions

The game is divided into several functions, each in its own file:

- gameMain
    - Handles the main game loop, physics, and display. It calls all other files for various tasks.
    - action()
        - This is the code called on every frame. Everything that occurs when the game is running is within this function. Everything within gameMain outside of this function exists to set up variables for this function.
- bufferKeys
    - This is called whenever the user presses a key (immediately, without waiting for the main game loop). It stores a value in the rocket's key buffer that can be read later.
- Const
    - This stores all constant values for the game, such as gravity, the mass of propellant to start with, and the sizes and positions of gauges, along with many others.
- spawnCow.
    - This is called whenever a cow should appear on screen. It handles enabling the cow and moving it to the appropriate location on screen, as well as resetting its timer to spawn again.
- ExhaustMgr
    - Manages exhaust particles, scrolling them and checking their lifetime along with everything else relevant to them.

Game Variables

The game has a lot of variables. A list of every single variable in the game would be impractical. Here are the general categories of variables:

- Sprites
    - The game uses several Sprite objects representing on-screen sprites, such as the rocket, cow, or title screen. These are assigned properties such as Location and Angle that control their display.
- Rocket physics
    - The rocket has many variables. Its physics computation does not use the SpriteKit (which has no tools for physics other than collision checking). These use values of the rocket such as its thrust, current position, and angle, along with other temporary variables such as delta_pos, which stores the distance the rocket has travelled in the current frame.
- Cow handling
    - The cow has many variable dedicated to keeping track of its state and important information. cow.State, for example, tracks what state the cow is in. It can be 'off', 'tractor', 'on', or 'fly'. This determines the cow's behavior, such as whether or not it will be scrolled or interact with the rocket.
- Exhaust particles' data
    - Properties of the Sprites that represent the particles. These include Scale, Location, Angle, and age, among others.

- Constants
    - Stored in Const.m. Do not change and can be accessed by any function.