Need for Speed: Hacker's Trail

6.111 Final Project Proposal

Richard Chan, Calvin Chung, Fan Yang

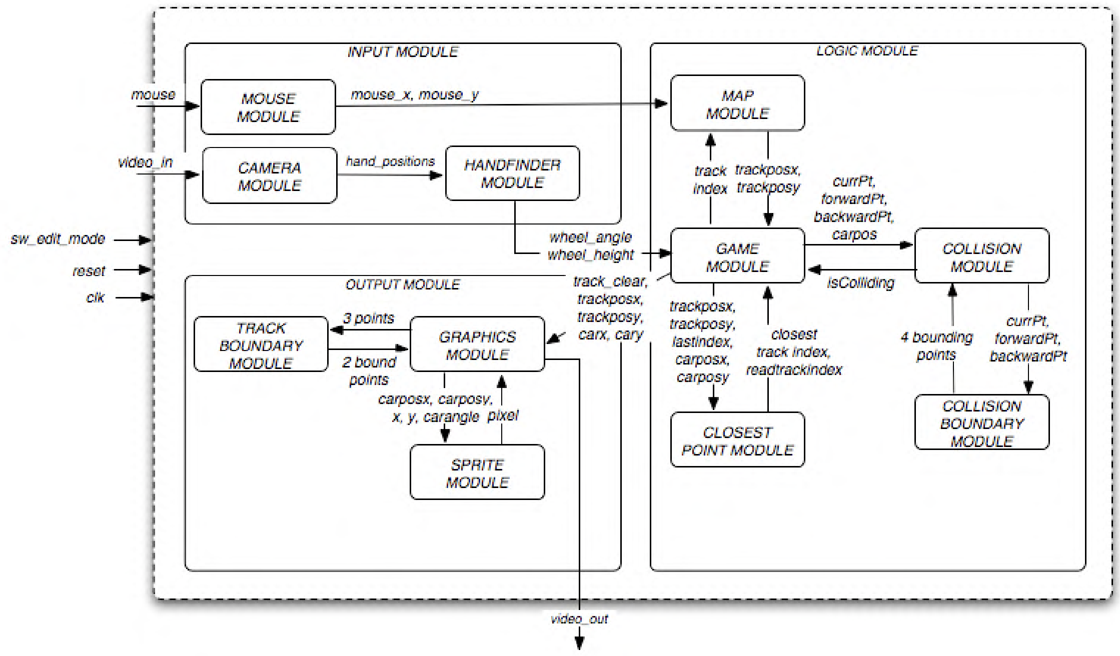
November 1, 2007

PROJECT OVERVIEW

The goal of the project is to implement a racing simulation system that allows two users to create their own race tracks and subsequently compete against each other on the created tracks. The users can create a desired race track by simply tracing it out using a mouse. The traced out tracks will be saved in memory and be loaded as the track for the game. Instead of using a typical wired steering wheel, each user will have an "air wheel" that can detect his motion and adjust both the direction and the speed of car accordingly. More specifically, the user will wear two colored gloves that will be used in conjunction with a video camera so that the location and motion of his hands can be detected.

The system runs in two modes (edit mode and game play mode) and can be divided into three major components (Input modules, logic modules and graphic modules). In edit mode, the input modules continuously take in inputs from the mouse and pass on the coordinates of the cursor to the logic modules, which generate the track and store it in memory. Using the graphic modules, the track is displayed on the screen in real-time. In game play mode, the input modules take in signals from the video camera to determine the positions of the colored gloves. The data is then passed onto the game logic modules, which control how the car will move based on the stored direction and velocity as well as the acceleration and change in direction specified by the inputs. The logic modules will also encapsulate the logic of collision detection. The graphics modules will then display the on-going game on the screen.

BLOCK DIAGRAM



INPUT MODULES

The main function of the input modules is to extract useful information from the stream of incoming input signals for use by the logic and graphic modules. The input modules consist of three sub-modules, namely the mouse module, the camera module and the hand-finder module.

Mouse module

In edit mode, we allow users to create their own racing tracks by moving the cursor of the mouse. The mouse module interfaces to a mouse connected to the labkit's PS/2 port. At every positive edge of the clock the module receives data input from the mouse and provides the map module within the logic modules with the x-y coordinates of the mouse cursor.

Camera module & Hand-finder Module

In game play mode, the user controls the speed and the direction of his car by moving the wireless "air wheel". To control the direction of the car, the user steers the wheel just like he would when driving a car. The speed of the car, on the other hand, is determined by the height of the wheel, which simulates the gas and brake pedals of the vehicle. As the user lifts the wheel higher, the car will accelerate, and the car decelerates as he drops the wheel.

To detect the motion of the wheel, the user will have to wear colored gloves and steer the wheel in front of a camera. The camera continuously sends image data into the camera module. The camera module is responsible for detecting the positions of the two gloves by running a color-matching and center-of-mass algorithm. The positions of the two gloves are then passed onto the hand-finder module. Within the hand-finder module, the average heights of the gloves are used to determine the height of the wheel. The difference in height between the two gloves, on the other hand, determines the angle of the wheel with respect to the vertical axis, i.e. the degree to which the wheel is turned. These two outputs, the wheel angel and the wheel height, will then be passed onto the logic modules.

LOGIC MODULES

Map Module

This module contains a RAM that stores the location information (x, y) of all the points on the track (by pixel), in order of how the track was drawn. During the edit mode, the map module will detect changes in the mouse inputs being sent from the mouse module and write each track point to the RAM sequentially. During the game play mode, this module will allow the game module to efficiently retrieve the location of a track point by using the memory address (an index starting from 0 representing the start point) of the track point.

Game Module

The game module will contain the current state of the game by storing the location, speed, and angle information of the car. The game module takes in the wheel angle and wheel height and calculates the change in velocity and rotation angle accordingly (the wheel height relative to the middle of the screen specifies the magnitude of the acceleration/deceleration). The results of this calculation will then be used to determine the next location of the car after one time step by explicit Euler integration. By continuously calling the closest point module, the game module can keep track of the closest track point (local minimum) to the car. The game module will also be able to detect collisions (at the next time step) by calling the collision module, such that, when collided, adjust the velocity of the car appropriately to prevent collision.

Closest Point Module

The closest point module will contain the logic of finding the track point that’s closest to the car given the closest track point of the last time step and the current location of the car. The module will start from the previous closest track point and evaluate the distances from the current car location to the adjacent track points one by one (looking ahead by a few points), in the order in which the track was drawn and also in the opposite order (in case the car starts going backwards), until the closest point is determined. This can be done by looking at the successive distances and stop searching when the distance is increasing and simply return the point that has the smallest distance.

Collision Module

Given the location of the car and the closest track point and its two adjacent track points, the collision module will determine whether or not there's a collision between the car and the boundary of the racing track. To get the bounding box (left and right bounds) around the car, the collision module can call the collision boundary module, which takes in the three track points and returns the four boundary points, two for each of the two boundary lines. Using these boundary points, the boundary lines can be constructed and used to determine whether or not a collision will occur. The collision module will then return an isColliding signal back to the game module to indicate whether or not a collision occurs at the current location of the car.

Collision Boundary Module

This module takes in the three track points closest to the car, the current point, the forward point, and the backward point (back and forward with respect to the current point in the order in which the track was drawn), from the collision module and returns the four points that can be used to construct the two boundary lines surrounding the current location of the car. The boundary lines can be constructed by using the forward track point and the backward track point to interpolate the slope of the boundary, which should be approximately the slope of the line connecting the forward point and the backward point. These boundary line segments are one half of the track width away from the current point and should have the same length as the length of the line connecting the backward point and the forward point. Then the location of the four points is simply the location of the end of the two boundary line segments.

OUTPUT MODULES

Graphics Module

This is the core of the output modules. It takes in signals of the car's position and the track's positions. The track's positions will be sent in as a stream such that when track_clear is on, all the track positions will be cleared from the screen; otherwise, each track position sent in will cause the module to record the position of the track point. The track positions can then be sent into Track Boundary Module to compute where the left-most and right-most boundary point local to the current track point. These boundary points will be colored to indicate the shape of the track. The car will also be drawn to the screen by using the Sprite Module, where the car will be drawn to the position on the screen specified by the Logic Module.

Track Boundary Module

The module takes in 3 positions as inputs, a "forward" point and a "backward" point, along with the local track point itself – the "current" point. The left-most and right-most point local to the current track point can be done by first drawing a line between the forward and backward point; the normal of this line intersecting the current point will be where the boundaries are. Take the two points a set track-width away from the track position on this normal line and those two points will be the boundary points.

Sprite Module

This module is responsible for drawing the car as specified by the location and angles given. The shape of the car will be retrieved from the ROM and rotated by the angle given -- each point will be projected using a rotation matrix given the angle. (Note that it will likely need to have a lookup on Sin and Cosine values to compute the rotation matrices.)

TESTING

The individual modules will be tested individually by simulation to ensure that they work according to the specification integrating them into the system. For the modules in the input component such as the mouse module and camera module, testing could be performed by actually testing against the inputs and observing the outputs. For the modules in the logic component such as the main game module, a sample track can be created and stored in the RAM for simulations. Also the wheel angle and wheel height that would normally be generated by user actions can be created directly in the module for testing. This will allow us to test the game module independently of the functionalities of the input module. For testing the output component, the sample (x, y) points could be generated and passed into the graphics module to be displayed onto the screen. After all the modules are tested and verified to work independently, the modules are then integrated. This will allow the system to be then tested as a whole by playing the game.

DIVISION OF LABOR

Our short-term goal is to build a prototype with the basic functionalities. During this stage, Richard Chan will be focusing on the logic modules, while Fan Yang and Calvin Chung will work on the input and output modules respectively. Once we successfully build a prototype, we will move on to improve our system by adding a multiplayer mode, refining our image processing methods, enhancing our collision detection algorithms, producing finer graphics, etc. At this stage we should be re-evaluating the work needed for each module and possibly readjust the division of work as needed.