2022

New Rally Y

DESIGN DOCUMENT FOR CSEE 4840

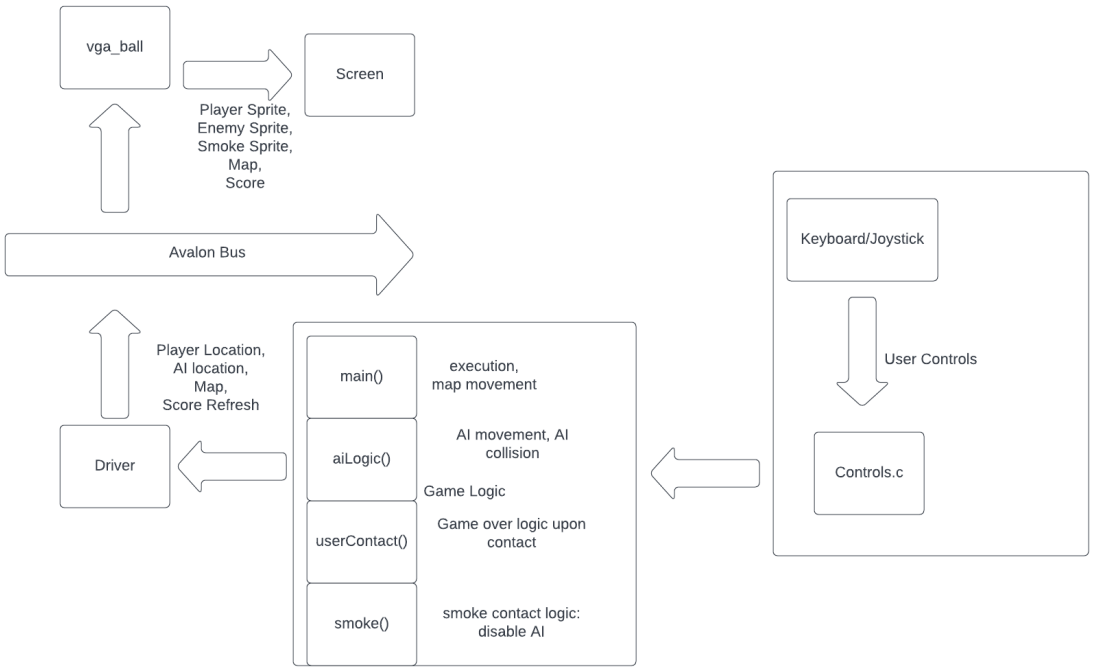
ANDREW JUANG

Content

1. Introduction

New Rally Y is going to be a tribute based on the video game "New Rally X". It is a classic retro video game where a "race car" sprite is controlled by the player via a joystick and directed through a set "maze" race track to collect flags for a high score while ai drive around either aimlessly or chase after the player. The player has the option to navigate through the maze to avoid them or use smoke to wipe out the enemies. Similarly, New Rally Y will try and emulate this game by having the player also control a target player sprite which can be controlled in 4 directions (up, down, left, right) to also gather flags in a less complex map maze. The AI will probably end up being dumb ai which will have two functions, either being a dumb wall which eliminates the player upon contact and is temporarily removed when the player uses smoke, or chases the player when they enter its proximity. The focus of this game will be mostly on the visual aspect, where sounds will be considered for later but sprite art and functionality will be the main focus.

2. System Block Diagram



3. Algorithms

Player Movement

Player movement will be decided using the input. Using a joystick or wasd to simulate a joystick, the player movement can go into any of the four cardinal directions up down left right. This will be done via similar logic to vga_ball, which was implemented in lab 3. The sprite shifts will be done similar to vga_ball. Input is tentatively via a keyboard and we might switch to a joystick for more similarities to the

original product, the keyboard will be the one provided by the lab. The player is restricted to a map which is boxed off by multiple walls, so collision detection must be added such that the player can only move in a restricted area. The original implementation used a zoomed in map, but for now we'll probably emulate a pac man map.

Enemy Movement

Enemy movement will spawn 3 enemies at the start which path randomly on the map. These should follow player restrictions, where collision detection is also preventing them from walking through walls. The enemies should follow a random path by choosing and trying one of the four directions every time they update, and when they are within a certain range of the player they can "home in" and start following them with varying degrees of consistency. Perhaps something along the lines of a weighted greedy randomizer where there is an n chance of following the player and a (1-n) chance to choose a random direction instead. If colliding with the player, it should result in a game over or a subtraction of lives if implemented

Sprite Generation

Sprite generation will follow the professor's lecture (https://www.youtube.com/watch?v=2ApafTCylus). Similarly we want to have a sprite attribute table with a name and tag. Based on the pattern map, we want to adjust the location of the sprite on the screen. We'd have each sprite for the player models and enemies have 4 different sprites for facing different directions. We'd want the map to be made of blocks so the obstacles could actually be 4 different variants as well to distinguish different varied obstacles of rectangles of different shape/orientation to simplify it. The static map or the zoomed in map with static player sprite both can use this feature.

Map Movement

If we have time, we will implement a moving map. This will be done via simulating a zoomed in environment, where instead of moving the "player" we'll move the entities around the player instead to simulate movement. This should be rather trivial to implement once the base map is resolved and coded in. The wasd or joystick will instead dictate where the "map" will "move" in context to the player.

Smoke Resource

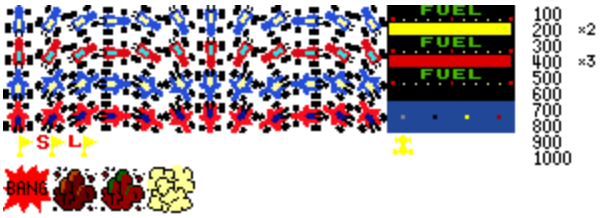
Smoke can be used via a simple button, which will make a "static" sprite appear. If interacting with an enemy it will temporarily remove said enemy for a set time, before "respawning" said enemy in a different location or perhaps in the same location as it was disabled to save data processnig. In this time, the enemy should have no collision box, show a different disable animation or be removed, and pause ai. This will have a resource timer so it cannot be spammed.

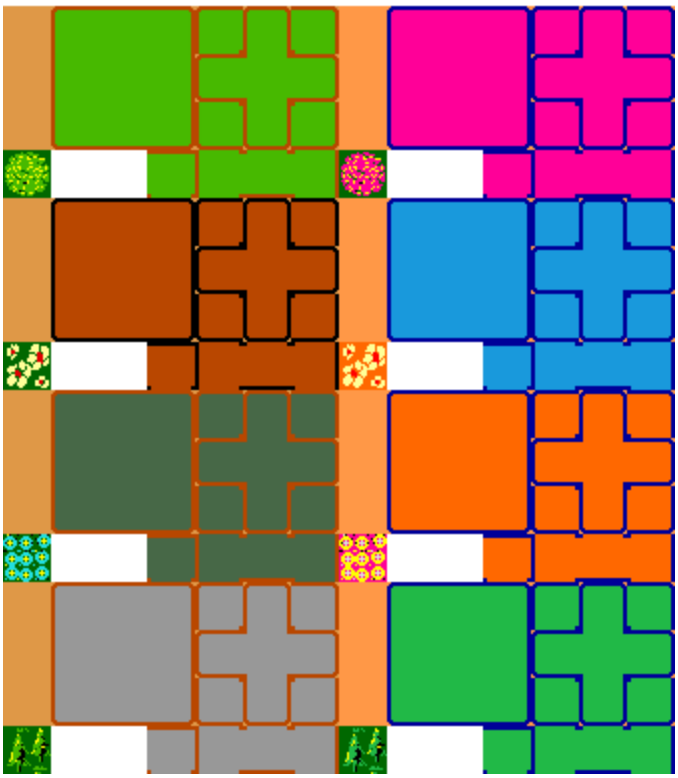
Flag Collection

Flag collection should be relatively simple. Upon contact with the flag sprite, the player should have their score increase. This should only register when the player hits the flag and also should spawn on a random part of the map. It should try to spawn anywhere on the map and redistribute if it hits a "wall" and the player cannot reach it there

4. Resource Budgets

Based on New Rally X sprites:





https://www.spriters-resource.com/arcade/rallyxnewrallyx/sheet/57635/

We want to simplify it down to a few main sprites

Category	Graphics	Size (bits)	# of images	Total size (bits)
Car Sprite		24*22	2 cars *4 directions	101,376
Smoke Sprite		16*16	1	6,144
Flag Sprite		16*16	1	6,144
Wall Sprite		90*20	2	86,400
Score Sprite		20*20	10	96,000
Game Status	GAME OVER	160*20	1	32,000
			Total Bits	328,064

Audio we don't want to consider at this original state since I'm not sure about the time for the implementation.

5. Hardware/Software Interface

Will be based on the hardware documentation found on the website:

http://www.cs.columbia.edu/~sedwards/papers/TMS9918.pdf

Will need to go to office hours to figure this out better.

Register 0: VDP option control bits

Register 1 (Contains Eight VDP Control Bits)

Bit 0 = selects 4/16k selection. This is relevant for our project

Bit 1 = Display Blank Enable/Disable. Will use to turn on and off the display, possibly after a game over

Bit 2 = IE (Interrupt Enable), might use when transitioning map screens, a slight pause before rendering

Bit 3,4 = Pattern Mode.

Bit 5 = Reserved Bit. Set to 0

Bit 6 = Sprite Size Selection, Probably will use Size 1 sprites (16x16)

Bit 7 = Sprite Magnification. Set to 0

Register 2: Register of Name Table sub-block

Register 3: Color implementation

Register 4: Base address of the pattern, text or multicolor generator sub block

Register 5: Base address of Sprite Attribute Table sub-block

Register 6: Base address of Sprite Pattern Generator Sub Block

Register 7: color code in text mode and color code for backdrop