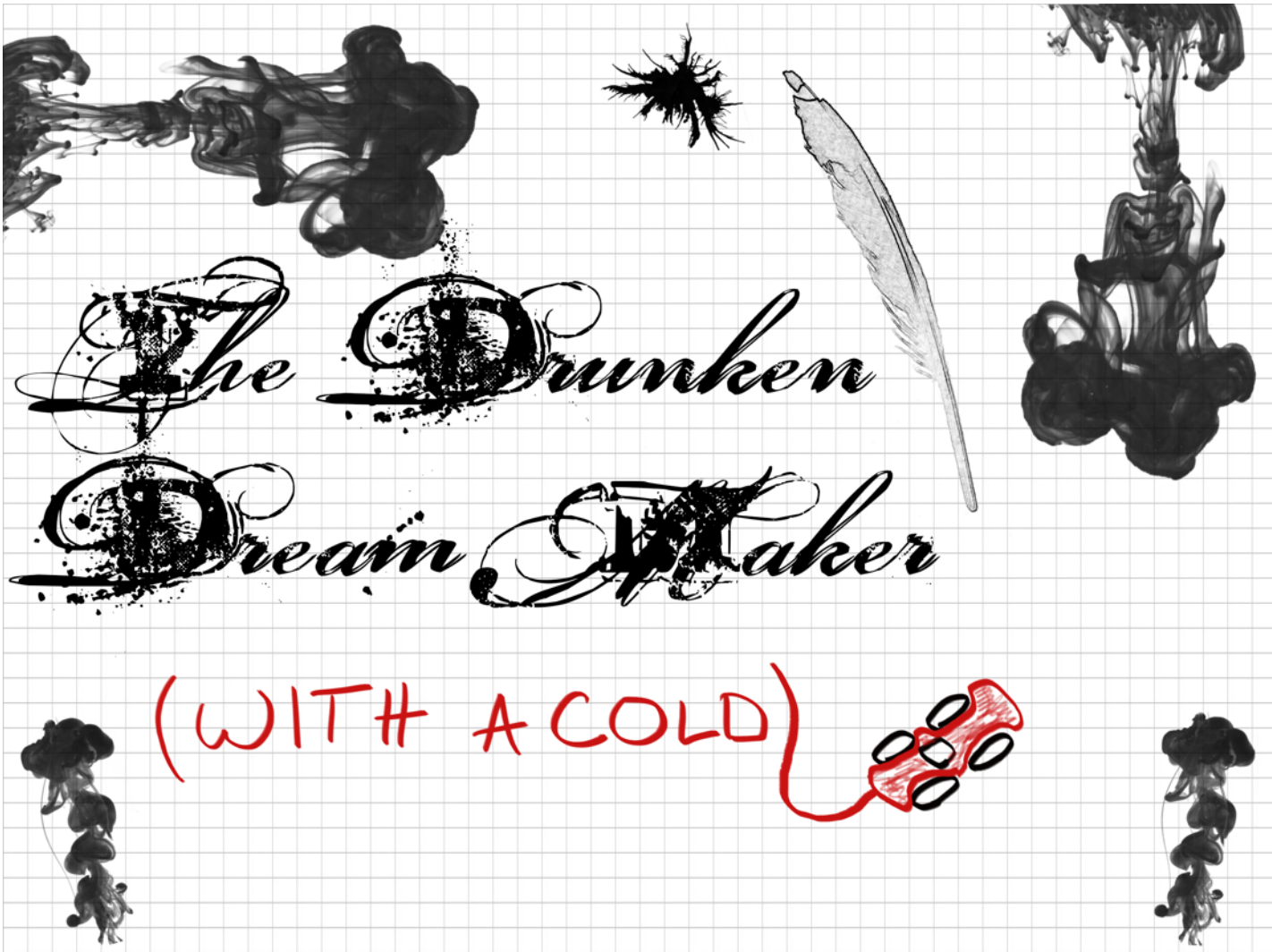
Triolozzi proudly presents



Riccardo Roveri, Vittorio Megaro & Emanuele Rudel

ETH Game Programming Laboratory
Spring Semester 2013

TABLE OF CONTENTS

Progress report

In the past month we developed the core functionalities of the game. We first setup the development environment with a GIT repository on Github and created the project containing the libraries of the physics engine. The next step was to discuss and sketch the architectural design of the game. After about one week we were ready to start the actual development of the game.

We managed to implement and test throughly all requirements of the *Functional minimum* and *Low Target* sections. At some point during this phase we realized that the game had a major flaw in the gameplay: the only race track became less and less predictable as we kept testing the game. This would definitely ruin the dynamism of the gameplay and not make it as much fun as we originally planned, therefore we decided to move the requirement of creating random tracks from the high to the desirable target. This is still a work in progress and we are happy with the results, but we will certainly improve on it during the next phase.

Gameplay

The core of the game consists of a racing game where up to four players challenge each other. The breakthrough idea of our game is to let players draw shapes and use them as obstacles to smite the opponents. In addition, closing a shape gives a temporary boost to the player as a reward. Beware of high speeds though, the car ABS (anti-lock braking system) might not work properly! Moreover while speeding up, the car does not draw and this introduces a challenge for players, that can either exploit or suffer the consequences.

As described in the game proposal, sticky notes lay around the track. The drawings on those notes determine whether they are dreams or nightmares. Players are awarded points when crossing dreams and lose points when crossing nightmares.

Although players will probably not realize it, the implementation of the camera was one of the most challenging tasks we had to face. We first adopted a split screen solution and quickly realized that it did not reflect at all our intent to bring people together and make them interact.

The camera now moves following the first player in the race and those who fall off screen are eliminated from the race. The difficult part of implementing this behavior was smoothing the camera movements as players were getting eliminated or the current ranking state changed. When a player is eliminated, the respective kid in the HUD wakes up and the score bar decreases a little.

We have received very positive feedback from different testers as they found the game very fun to play, although they pointed out that the driving controls are quite difficult for beginners. For this reason we have developed a number of driving modes and we are still not sure which one we would like to keep.

Graphics

The game features a simplistic doodle graphics since it takes place on a storyboard. We just started drawing the basic elements and we plan to improve the graphics a lot during the last development phase, as it rather easy to replace the existing resources with new ones.

We already use simple shaders to draw all the game elements in order to take advantage of the GPU power. In the spare time we developed a simple fluid simulation (in a separate project) that runs on the GPU and we plan to use it for different special effects in our game.

Challenges

As already mentioned before, developing the camera feature was one of the most difficult tasks up to now. The camera behavior and scoring systems also adapt automatically depending on the number of players (from 2 to 4) and this proved to be quite challenging.

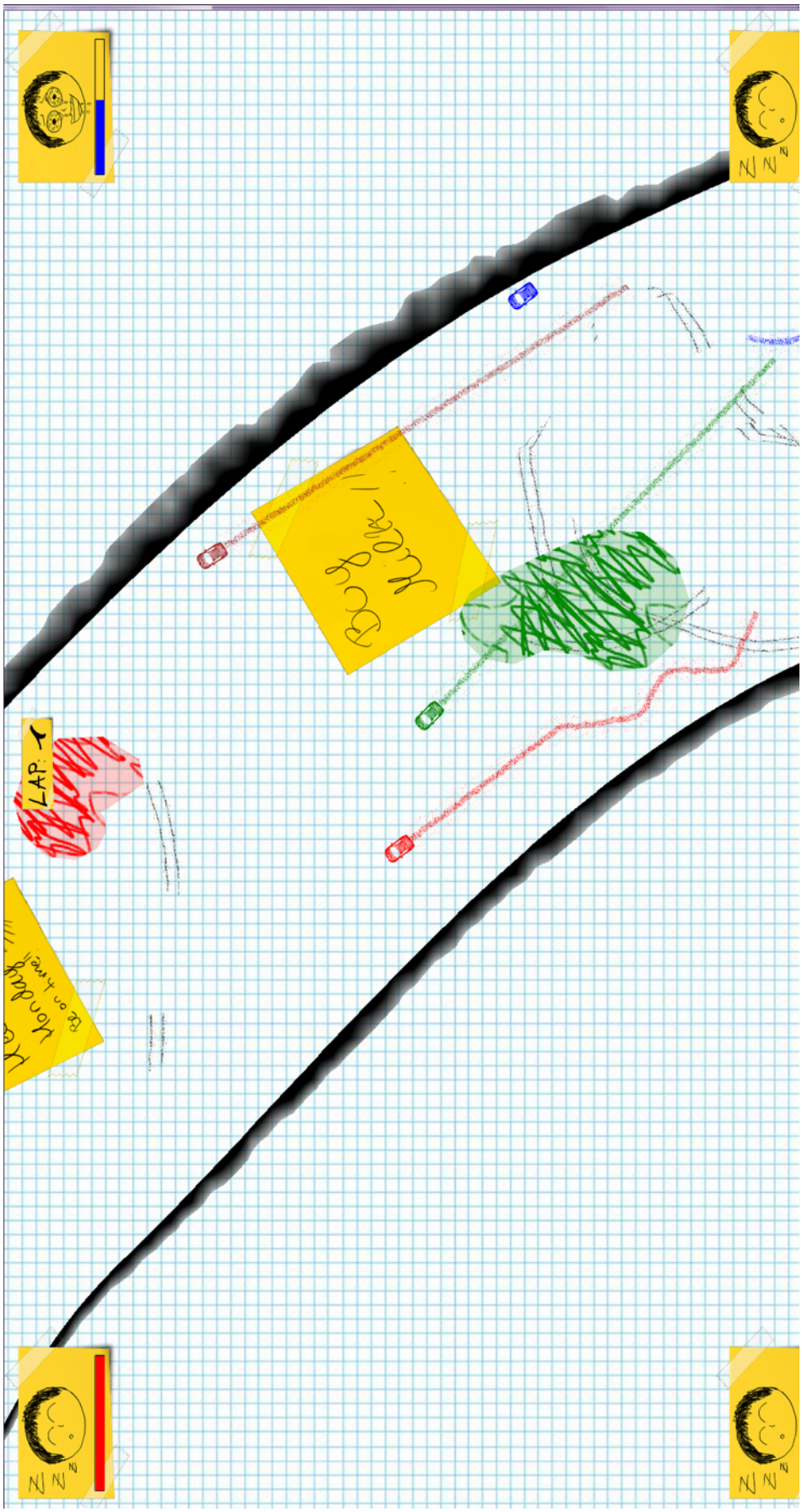
Another challenge was the generation of random tracks. The difficult part here is not to randomize the tracks but to make them fun. We are still looking for a pattern that achieves this goal.

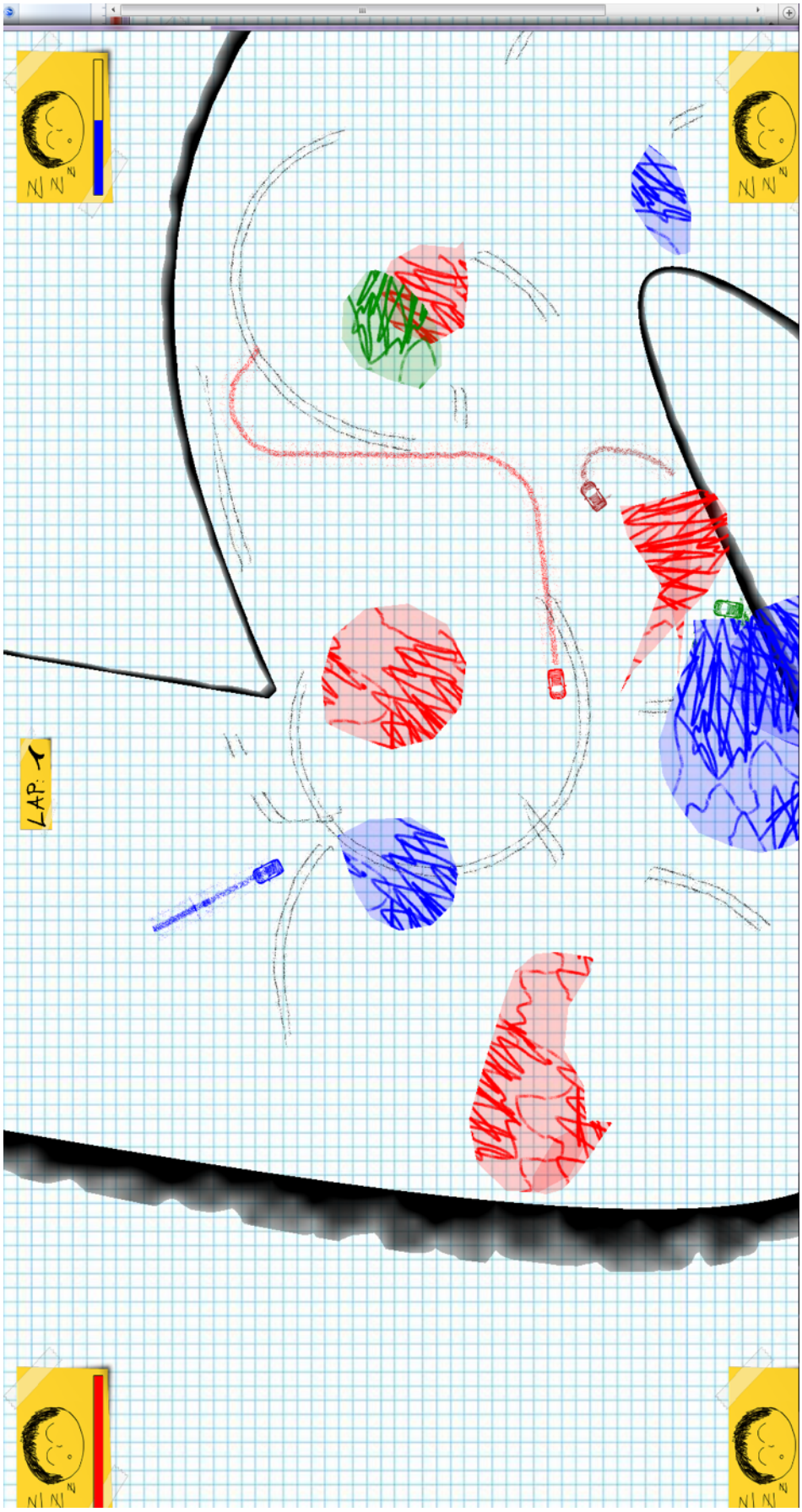
We encountered a few problems with the XNA framework since we found the documentation to be not very thorough. Many documents and tutorials were also written for older versions of the framework and were not useful. Moreover, we thought about using the Shader Model 4.0 for advanced computations (e.g. track generation and fluid simulation) but later discovered that it is not supported by the latest XNA version.

Our game uses the Farseer physics engine to detect and handle collisions and to model the cars on the track. The advantage of using this engine is that it is easier and faster to setup the objects in the game world. The lack of documentation, however, gave us a few headaches as we struggled to understand the basics during the first development weeks.

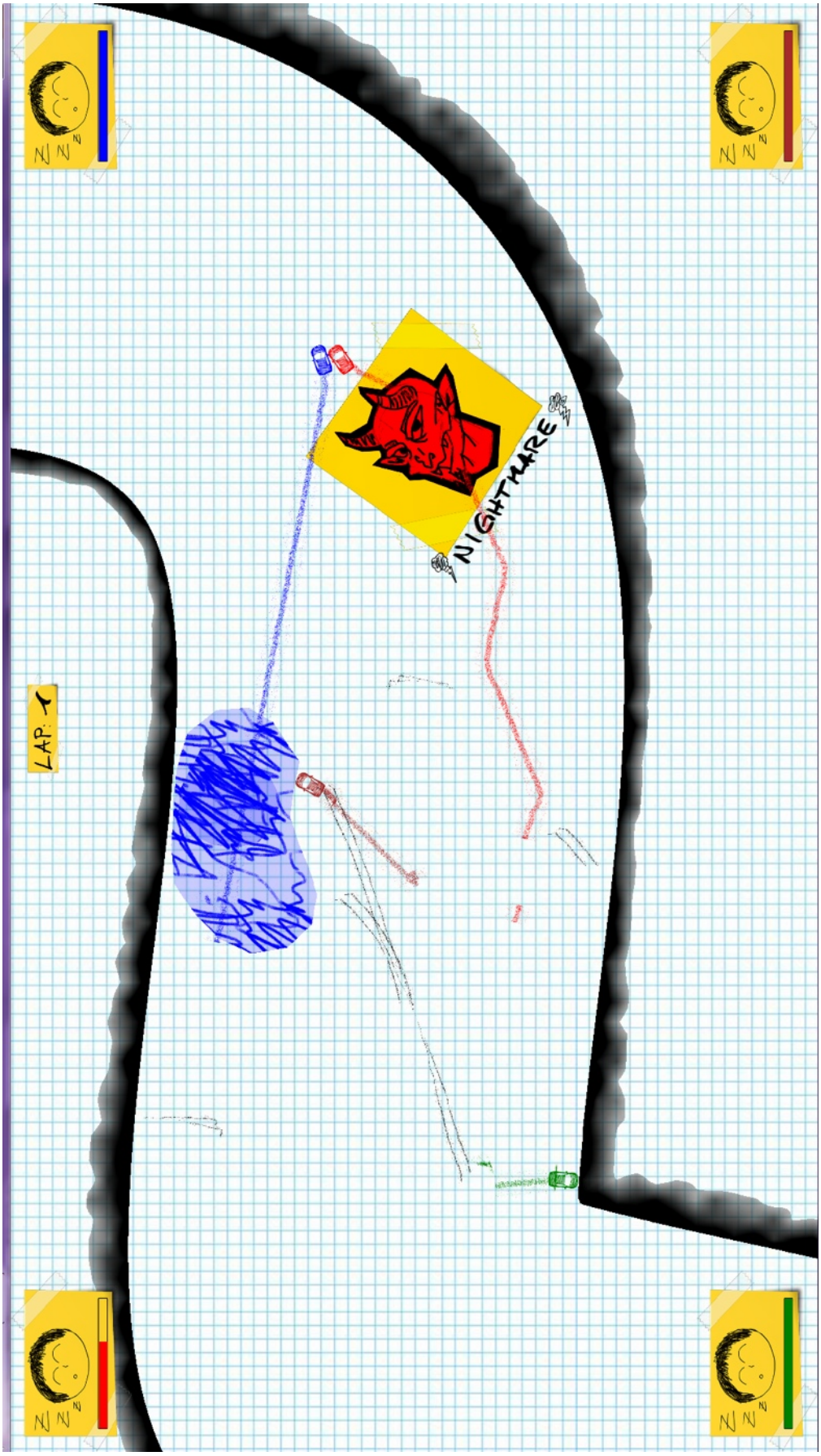
As suggested during the lectures, we developed the game always keeping in mind the performances issues related to memory management. We used a lot the profiler on the Xbox and made sure that no lagging would occur during the racing game.

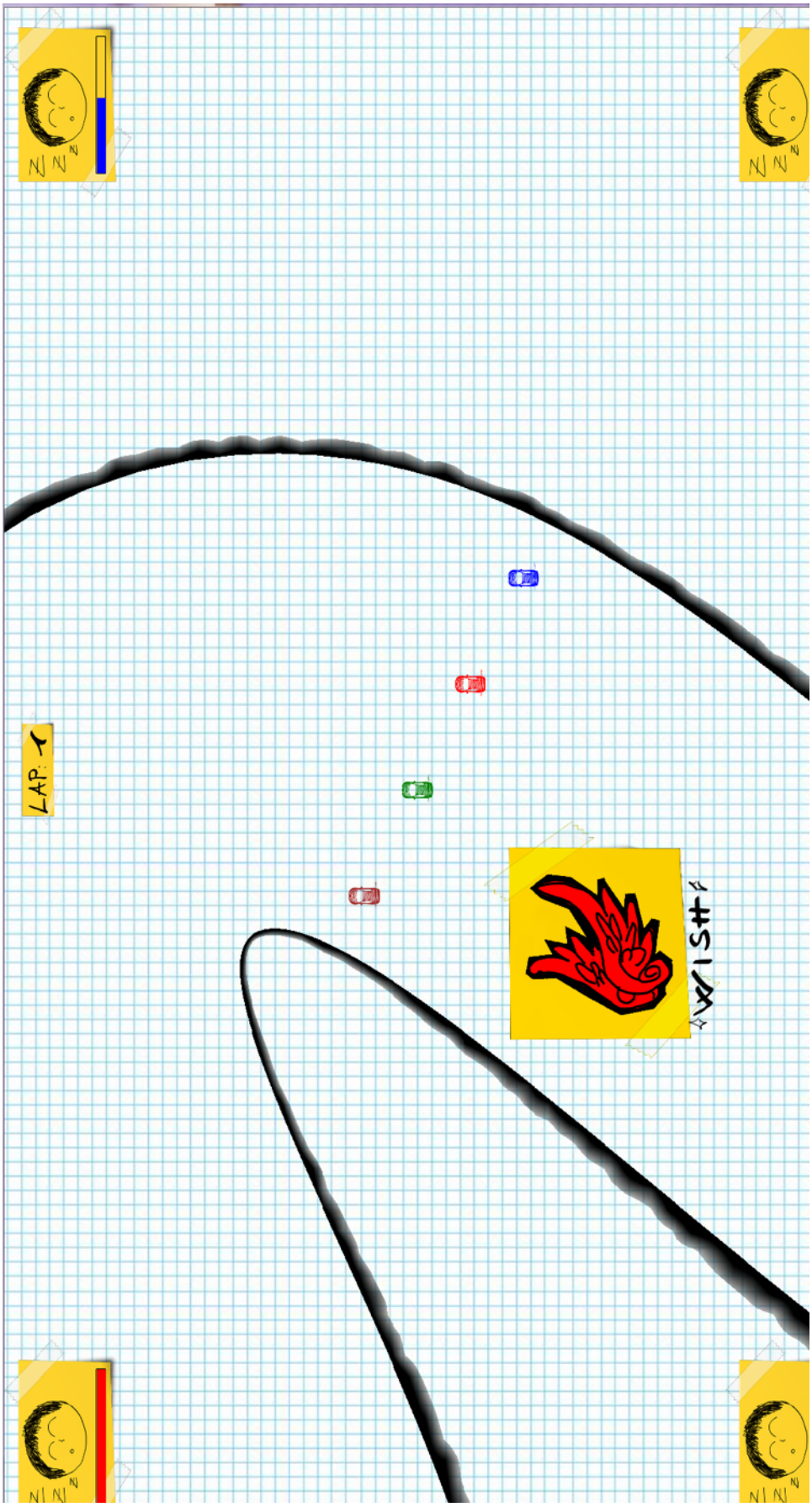
Screenshots

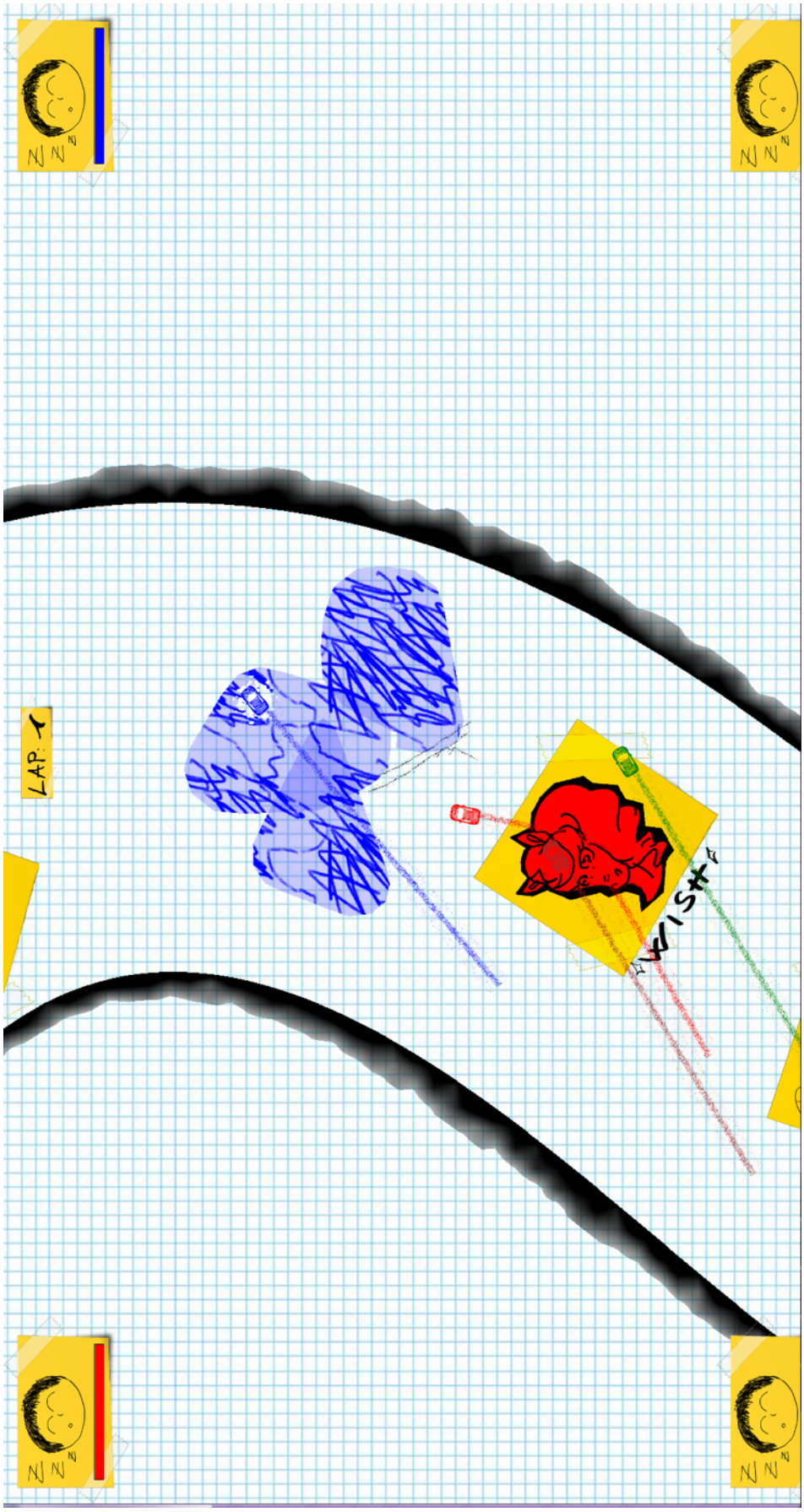

LAP: 2


LAP: ۲

LAP: ٢
NIGHTMARE


NIGHTMARE
LAP. 1


LAP: ٢


LAP: 2

Development Schedule

Functional Minimum

Task ID	Description	Status
11	Racing game only	COMPLETED
12	User input handling	
13	Camera system	
14	Simple scores	
15	Single, fixed race track	
16	Basic collision detection	
17	Basic graphics	

Low target

Task ID	Description	Status
18	Simple drawing (trails) with boost effect	COMPLETED
19	Obstacles	
20	Game menu	
21	Nice graphics	

Desirable target

Task ID	Description	Status
22	Advanced scores	Pending
23	Accurate drawing	Pending
24	Different sleeping phases	Pending
25	Background music, sound effects	Pending
26	HUD	Partially completed

High target

Task ID	Description	Status
27	Race track procedurally computed online	Partially completed
28	Ink/pencil effects	Pending
29	Procedural drawing effects	Pending
30	Nice transitions between phases	Pending
31	Introduction of mucus as an obstacle	Pending
32	Deformable objects	Pending
33	Particle effects	Pending