Iron and Feather

(Working title)
Interim report

Urs Dönni
Martin Seiler
Julian Tschannen

April 14, 2008

Contents

1 Gameplay **2**
1.1 Steering 2
1.2 Playfield 2
1.3 Collision detection 2

2 Fluid Simulation **3**

3 Graphics **3**
3.1 Models 3
3.2 Textures 5
3.3 Particles 5
3.4 Fish flocks 5
3.5 Blur 5
3.6 Grass bundles 5

4 Other **5**
4.1 Artificial Intelligence 5
4.2 Sound 6

5 Development schedule **6**

6 Problems **6**

7 Design revisions **6**

1 Gameplay

1.1 Steering

The prototype for the gameplay mainly served for the purpose of finding good steering for the ship and the turret with the Xbox controller. Since the prototype we have added one more way to control the turret: moving the right thumb-stick in a direction immediately shoots into that direction without having to press an additional button (see Figure 1). Which controls we will choose will still have to be determined, the new controls seem to work pretty well though.

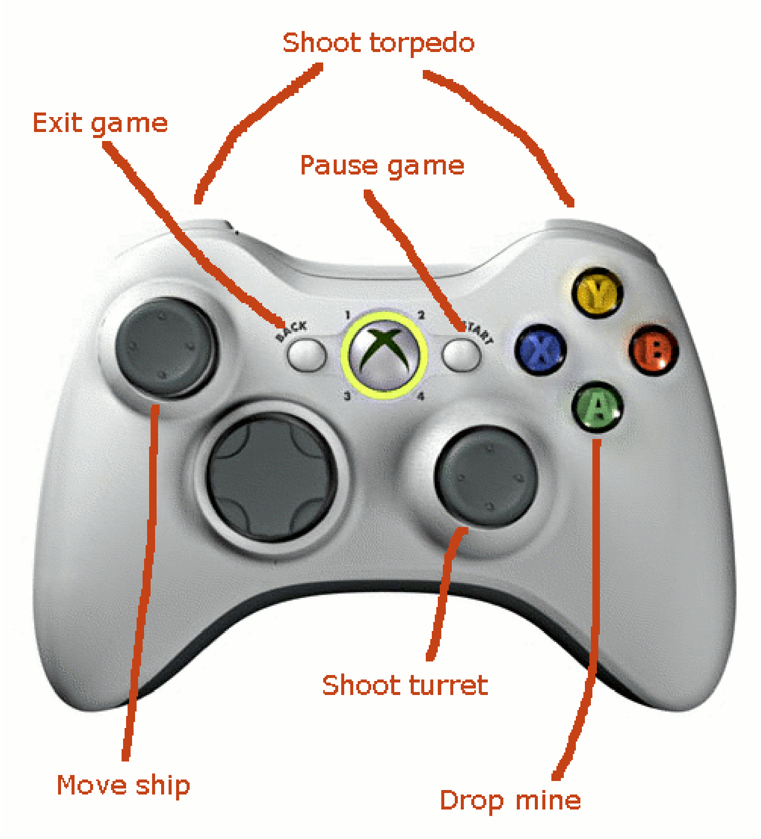


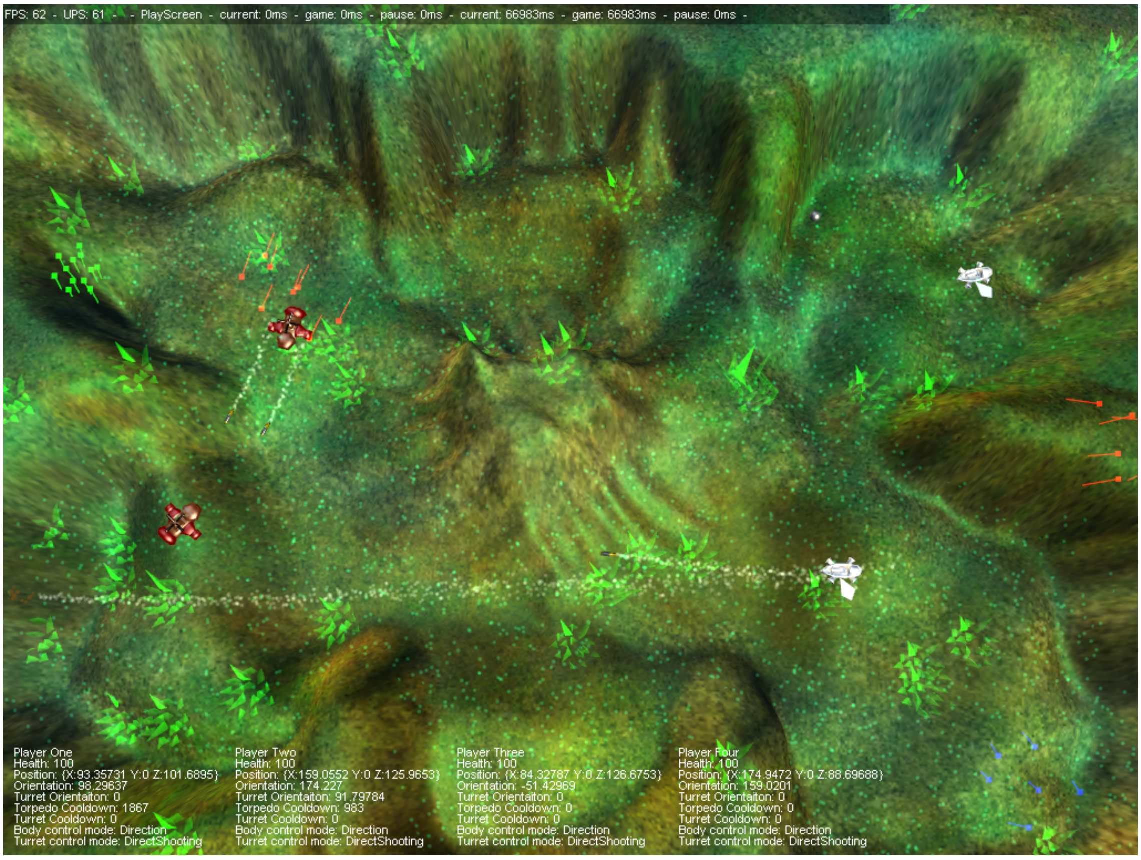
Figure 1: The new control scheme.

1.2 Playfield

We have created a test level on which the game can be played. Using a heightmap, we generate a terrain. The ships can drive around in the valleys of the terrain (see Figure 2).

1.3 Collision detection

Collision detection has been implemented, Therefore the game is now really playable for the first time. Multiple players can navigate in the playfield, they cannot drive through walls, they can shoot each other and the shots are blocked by walls.



```
FPS: 62 - UPS: 61 -  - PlayScreen - current: 0ms - game: 0ms - pause: 0ms - current: 66983ms - game: 66983ms - pause: 0ms -
```

```
Player One
Health: 100
Position: {X:93.35731 Y:0 Z:101.6895}
Orientation: 98.29637
Turret Orientaiton: 0
Torpedo Cooldown: 1867
Turret Cooldown: 0
Body control mode: Direction
Turret control mode: DirectShooting
```

```
Player Two
Health: 100
Position: {X:159.0552 Y:0 Z:125.9653}
Orientation: 174.227
Turret Orientaiton: 91.79784
Torpedo Cooldown: 983
Turret Cooldown: 0
Body control mode: Direction
Turret control mode: DirectShooting
```

```
Player Three
Health: 100
Position: {X:84.32787 Y:0 Z:126.6753}
Orientation: -51.42969
Turret Orientaiton: 0
Torpedo Cooldown: 0
Turret Cooldown: 0
Body control mode: Direction
Turret control mode: DirectShooting
```

```
Player Four
Health: 100
Position: {X:174.9472 Y:0 Z:88.69688}
Orientation: 159.0201
Turret Orientaiton: 0
Torpedo Cooldown: 0
Turret Cooldown: 0
Body control mode: Direction
Turret control mode: DirectShooting
```

Figure 2: Current look of the playfield. The game is playable at this stage.

2 Fluid Simulation

As described in the report about our prototypes, the fluid simulation was a major performance issue. Therefore we planned to implement the simulation on the GPU. We have finished the implementation and are pleased with the result. Whereas we couldn't simulate a 50x50 grid with the CPU implementation, we now can now easily simulate a 128x128 with much higher accuracy (10 times as many iterations when solving the linear systems) with no performance issues on the PC or the xbox.

We have integrated the simulation into the game. The ships and shots react are influenced by the fluid and vice versa. Figure 3 shows the velocity field and the boundaries (which can be arbitrary) as an overlay.

3 Graphics

3.1 Models

So far we have created two models of submarines, both of which are integrated in the game already. The two submarines are called Hummingbird (Figure 4) and Interceptor (Figure 5). Further we have two types of shots and one kind of mine.

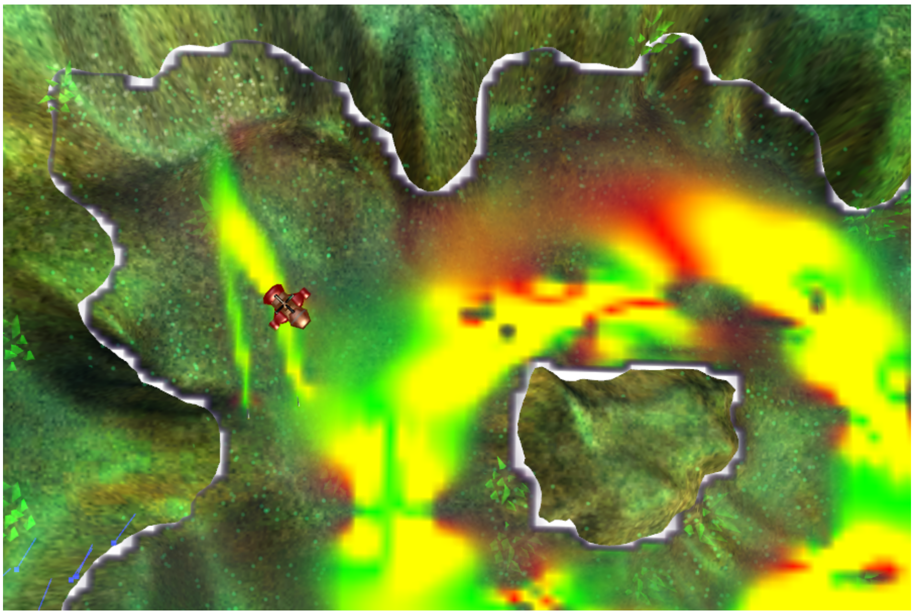


Figure 3: Playfield with an overlay showing the boundaries and the velocity field of the fluid simulation.

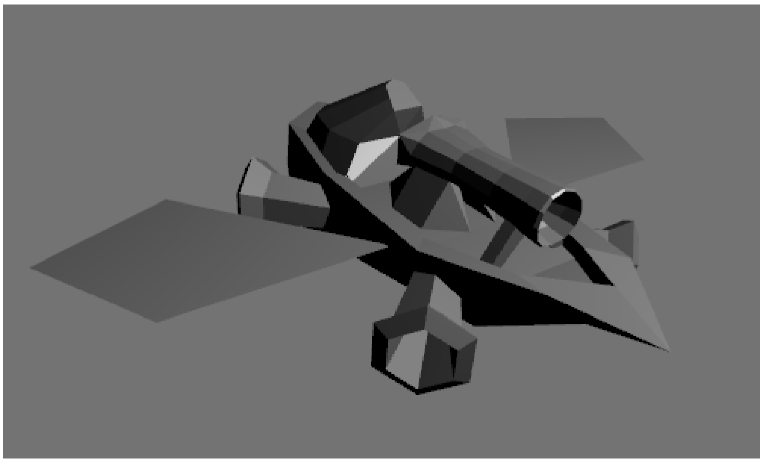


Figure 4: Hummingbird submarine

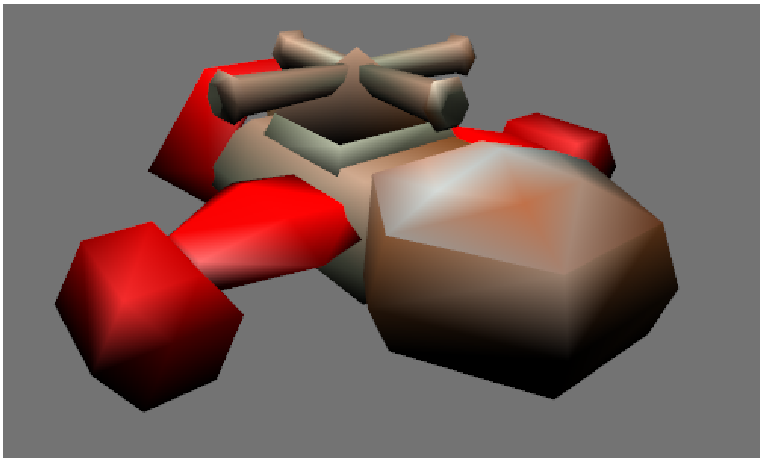


Figure 5: Interceptor submarine

3.2 Textures

The above described models are already textures. We also have a texture for the terrain.

3.3 Particles

We have implemented several particle systems for our game:

1. **Particle fluid** The fluid is made visible by throwing particles into it and having their movement be influenced by the fluid.
2. **Bubbles** Behind the ships and the shots we generate bubbles.
3. **Dirt** When an object hits the wall, some dirt is swirled up.

After we implemented the particle systems, we discovered that we have performance issues with lots of particles on the xbox. For this reason we started to implement the particle systems on the GPU. This is not yet finished.
We believe that using the GPU to calculate the fluid simulation and the particles, we'll have removed all bottlenecks we'll encounter on the xbox and from now on can concentrate more on content and gameplay.

3.4 Fish flocks

We implemented a flocking algorithm to simulate fish flocks swimming through in the water. The fish are not part of the gameplay, but they avoid ships, shots and walls.

3.5 Blur

To make the fluid more visible, we implemented a blur effect for the fluid, which blurs the playfield in areas of high fluid velocity.

3.6 Grass bundles

All over the playfield are small bundles of grass which move with the fluid. This improves the realism of the game and makes it clearer that the scene is under water.

4 Other

4.1 Artificial Intelligence

Enemies for now only shoot in the direction of other ships, there's no other AI yet.

4.2 Sound

It's possible to play sounds when something happens in the game. The only sound in the game for now is now is when a shot is fired.

5 Development schedule

Apart from the AI, we have finished implementing layer two of our development schedule. We also have finished parts of layer 3 (models, textures, nice looking game).

6 Problems

We had major problems with the implementation of the fluid simulation on the GPU. In the end, the biggest two issues turned out to be the following:

> On the Xbox, floating point textures (SurfaceFormat: HalfVector4) can only be accessed using point sampling. There's no warning, but accessing the texture using anything else than pointsampling produces garbage in the output.
>
> *XNA* has a buggy conversion from Vector4 to HalfVector4. On the Xbox, a HalfVector4 is represented as (w, z, y, x), *XNA* however doesn't do the conversion correctly on the Xbox.

The implementation of the particle system on the GPU also proved harder than expected. Our particles have to react to the fluid simulation, which makes the implementation trickier than existing implementations we found.

We switched from Maya to Blender after two weeks of trials with Maya and a couple of destroyed models due to bugs in Maya. Blender works fine so far for us.

Because we spent more time than scheduled on these two problems, we are a bit behind schedule. We have less models than we expected to have, we don't have a level editor yet and we don't have many sounds and a working menu yet.

7 Design revisions

We don't have big changes on our goals. The only thing we dropped is that the player can buy more weapons and and shields. We believe that this doest not fit the style of the game, which should be fast and uncomplicated. However, we recognize that this would have increased the deepness of the game on the long run.