

Final Report – Game Programming Lab `07

Group members: Benjamin Schindler, Basil Fierz, Henning Avenhaus

Gravity Bound

Introduction

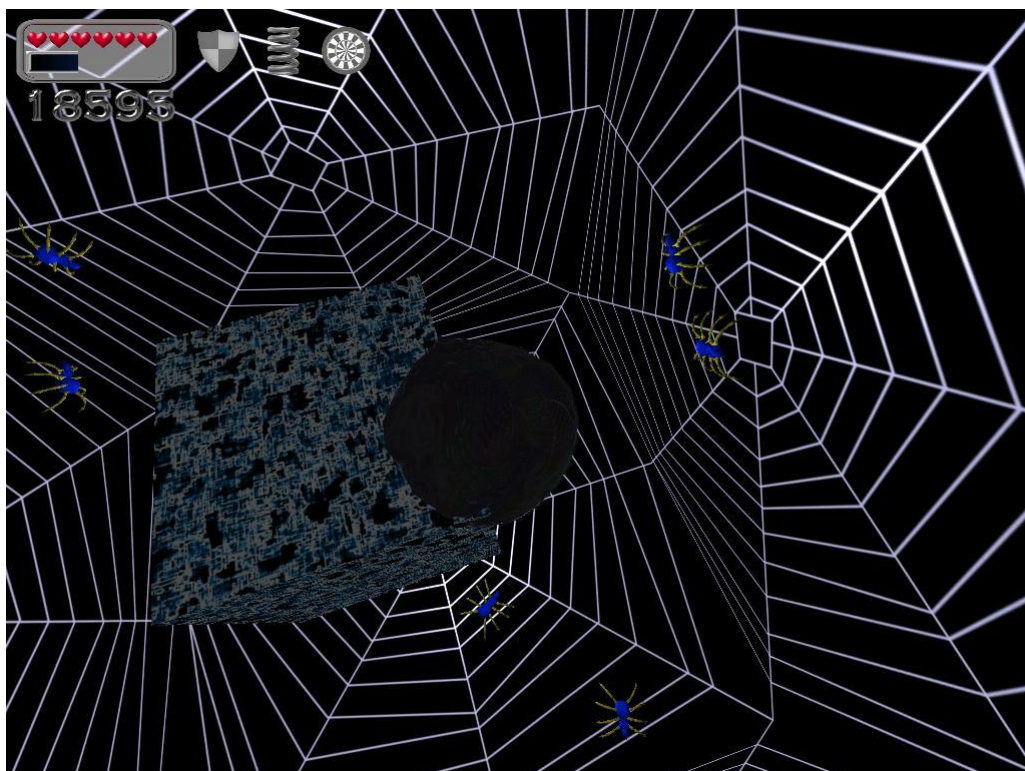
The game is an arcade style game, which orients itself loosely on the old and famous arcade game „Bubble Bobble“ (1986) and a blob-like game called “Gish” produced in 2004.

“Gravity bound” is about a heroic drop of oil, named “Blob”. He lives in a three-dimensional cubic world, which can be turned around to switch the direction of gravity. Though it is not completely clear where he comes from, there are certain hints. There is a horror movie called “The Blob”, created in 1958. He has several possibilities to move. He can “slime” forward, jump, crawl up walls and obstacles and he can slide over the ground. He is also able to stick to walls.

But Blob doesn’t live alone. He has to share his home with monsters, which try to kill him. To find his peace again Blob has to defend his world by chasing and killing the monsters. Fortunately he is not unarmed. Blob can shoot oil bullets to paralyze his opponents. The ammunition, which he shoots, comes from itself. The more Blob shoots, the more he shrinks. If he shrinks below a certain volume he can’t shoot anymore and thus is unprotected further on.

Monsters Blob has paralyzed, he can eat and devour. If the monsters by any chance are not paralyzed and Blob touches them he loses a life.

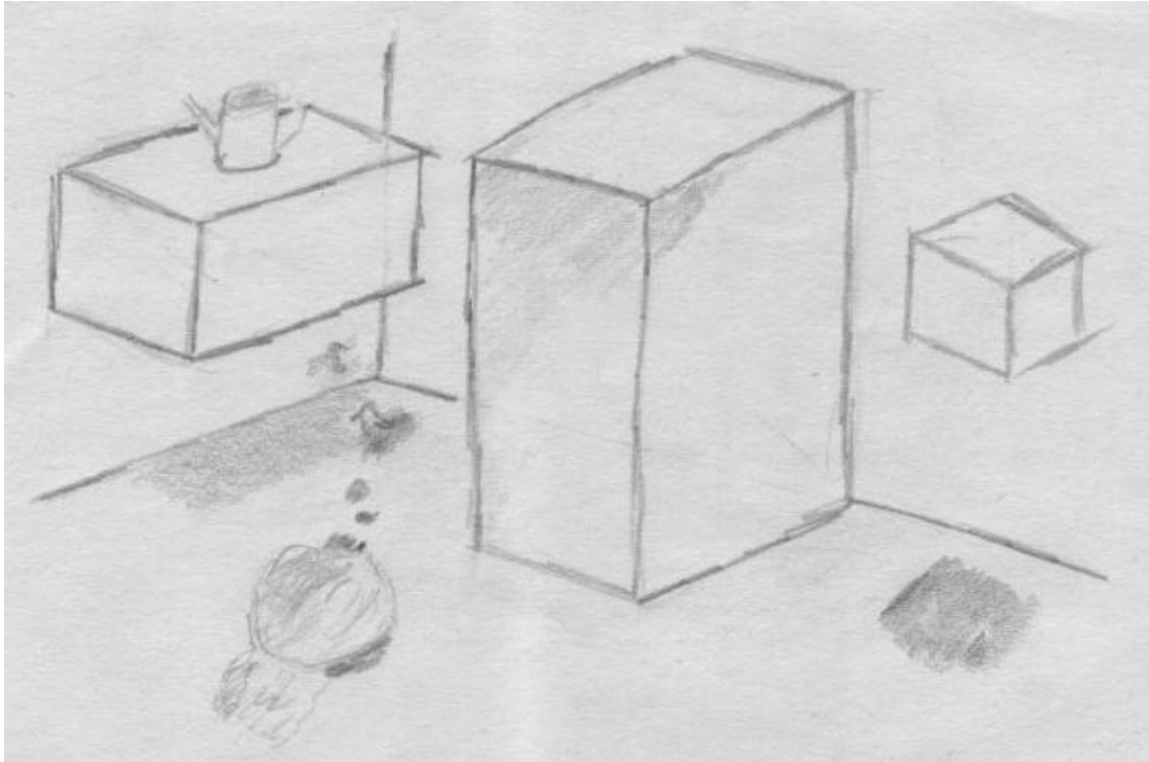
The blob itself creates a very unique feel for the game. The ability to switch the gravity and the blob’s ability to stick to walls combined create unique strategic possibilities. For example, in the 4th Level, the only real way to survive is to stick to the central cube and to snipe down the spiders as seen in the screenshot.



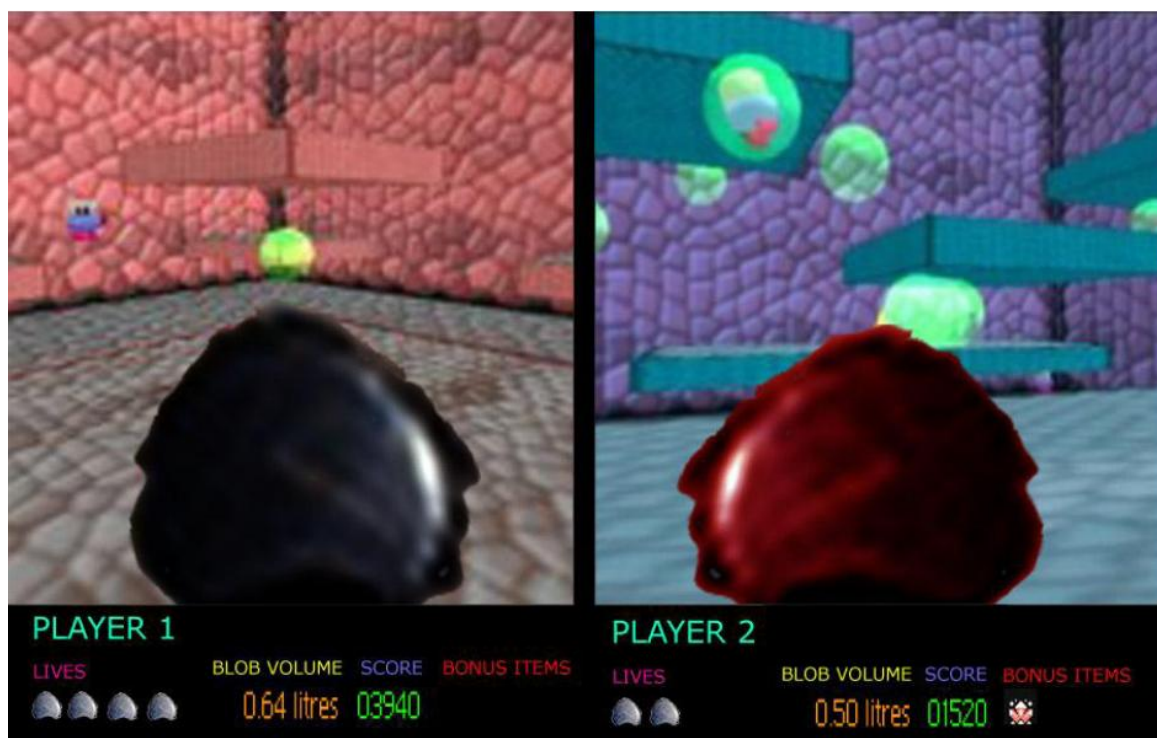
Design

Concept Artwork

This is our very first screenshot of our game. We used a pencil-shading technique to produce it.



The following sketches were done in Photoshop. They show the initial concept for the OSD and the split screen multiplayer.



It should be noted, that the game produced looks remarkably close to the mockups we created.

Game Play Details

To make the game more appealing for non Blobs, it is designed as a 3D arcade style game. The goal of “Gravity bound” is to score points. The more you score, the better you are. This can be achieved with two means: either by killing enemies or by collecting bonus items.

Blob has more than one life and consists of a variable amount of oil, depending on the amount of shots he has emitted and the amount of oil he has collected through bonus items. He also gains volume by digesting eaten monsters.

“Gravity bound”, as the name says, is about gravity. The player can tilt the box world by 90° degrees in each direction. Everything is affected by the new gravity: The blob, the monsters, the bonus items and the shots Blobs emits. Switching gravity really turns Blob’s world around.

Oil

As stated earlier the amount of oil of which Blob consists is variable. It reduces when he shoots and it increases when he eats monsters or collects bonus item. Blob can also recollect the ammunition he emitted by going over them. As a natural cause of this, the blob shrinks and grows with the volume he has. This also has an impact on the speed the blob can move which in turn affects strategy and game play.

Shooting

The Blob always shoots in his view direction. This way it is possible that he can also shoot upwards. The shots are affected by the gravity.

Bonus items

The game contains a few items, which Blob can collect:

- Oil cane: Add new oil to Blob.
- Shooting stabilizer: The shots are not affected by gravity.
- Life: Blob gains one additional life.
- Shield: Blob can touch a monster without dying one time. After touching a monster the shield vanishes. Blob stays protected for a few more seconds.
- Jump enhancer: Blob can jumper much higher.

Enemies

In his box world Blob has to face four types of enemies: Walking, jumping, flying ones and spider like monsters, which have the ability to walk across every wall without being affected by gravity. Blob loses one life if he touches a not paralyzed opponent. Different monsters move in different ways. If they start appearing in hordes, Blob really gets a problem and needs to move and shoot quickly.

Level elements

Two level elements were planned: trampolines and Beamers. However, due to the lack of time, none were implemented.

Score system

Scoring points is possible in several ways. The enemies Blob eats add points to the score. He can collect items, which also add points. At the end of each level the oil volume above the initial mark is also added to the score. Blob starts the new level with a standard volume. Because the score system is the most crucial part of the game (scores are used by players to compare each other), it is hard to

balance. Thus the final amount of points the player gets for an action were determined during play testing.

Multiplayer

As most arcade games “Gravity bound” can also be played in split screen multiplayer mode. We wanted to implement a four player Version, however, due to performance issues on both the Xbox and the PC, we reduced multiplayer to a two-player mode.

The players can collect not only the own shots and eat the monsters they paralyzed themselves, but also the ones from the other players.

As gravity switch in the multiplayer case affects all players in the same way this can cause a lot of confusion for the other player. However, play testing showed that it is relatively easy for the players to adapt to the new situation of changed gravity.

Controls

The game can only be played using an Xbox360 controller. The controls are similar to the ones of many 3D games. One steering cross is used to maneuver the main character (forward/backward/striving) and one to change the view direction.



Process

We did not do a lot of design on the way of creating the game. It was our opinion that we don't exactly know what we need in a few weeks. As a result, we did the coding in a very evolutionary way. There were quite a few parts which have been rewritten partly or entirely more than one time. But the advantage of this was that we had a very simple working version fast and redefined the design while actually coding new features.

Mass-Spring System

In order to make the blob itself a soft, deformable body, a mass-spring system was used.

The system consists of one mass point in the middle of the blob and a lot of mass points on its surface. The mass points on the surface are generated with an icosahedron (12 points, 20 faces, 30 edges), which is then tessellated twice to give 162 surface points, 320 surface triangles and 480 surface springs. Each point is then connected via a spring to the middle mass point.

To ensure (approximate) volume preservation, volume restraints were introduced for each of the 320 tetrahedra built by the points of one surface triangle and the point in the middle.

Damping was added to make the blob stable.

After fine-tuning the different parameters (masses separate for surface mass points and the point in the middle, stiffness for surface and internal springs, stiffness for volume restraints), the system was still not very stable and required small time steps. Several actions were taken to reduce this problem:

- A force-based damping was introduced and used in addition to the normal one
- Maximum values for spring and volume forces were introduced and used to avoid very high forces from making the system break down
- The damping was made dependent on how “heterogeneous” the velocities are, i.e. if the blob is heavily wobbling, the damping will be stronger to avoid breakdown
- The collision detection was refined to better cope with small objects you can collide with and avoid strange effects on corners sticking into the blob

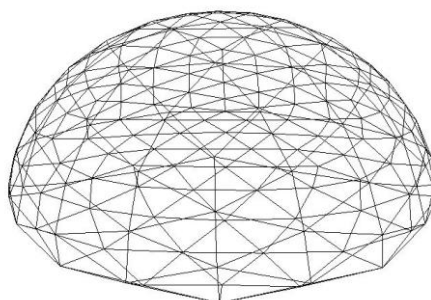
After all these, the blob was sufficiently stable and fast to simulate to give a smooth gaming impression (on fast PCs).

Furthermore, the system had to be able to cope with varying volume of the blob. This was done by scaling the rest lengths of the springs and the rest volumes for the volume constraints as well as scaling the stiffness factors and masses accordingly.

Movements are applied with a combination of rotational and linear forces (to make the blob roll over the ground, rather than slide on it).

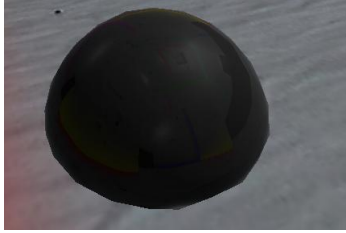
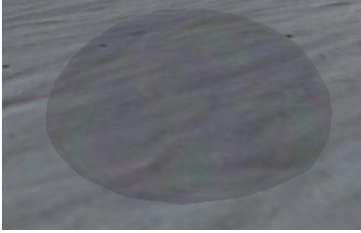
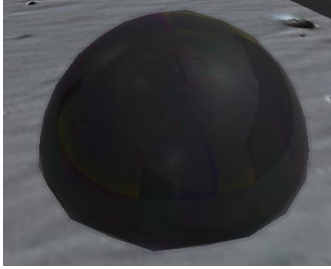
The stick feature produced more problems with instabilities, which could, however, be solved. Sticking is applied by fixing points once they collide with the wall. The mass points can then not move anymore until the stick feature is disabled.

Below is a picture of the mass-spring system (only surface springs displayed), lying on a white plane.



Rendering of the blob

To make the blob more appealing to the player, we tried to give it a realistic oily and shiny look. We used an environment map to mimic the reflective nature of the oil. To mimic the refractive part we used some fake rainbow colours.

		
The blob rendered with only the environment map.	The blob rendered only with the rainbow effect map.	The blob rendered with both effects blended over each other.

To visually improve the environment mapping we combined it with the standard Lambert lighting model and we added an approximate Fresnel falloff.

Development

Our development schedule was targeted to get a simple, yet playable version as soon as possible. With only minimum features implemented (no gravity change, blob as solid sphere, monsters as simple spheres, very basic monster AI, ...), we hoped to still grasp an early feel for the game and then start to gradually implement the additional features.

We tried to break down the development into parts.

Basil was responsible for all the graphics (including things like environment mapping), and the basic framework of the game. He also was responsible for running the game on the different platforms, for profiling and several optimizations later.

Benjamin was responsible for all the sound in the game. He also developed the collision detection, the input system and most of the game logic.

Henning was the one caring about the blob and its mass-spring system. He also did most of the work on monsters, like monster AI (movements and rotations), and the levels and textures.

With this kind of splitting work, Benjamin still does not know how the mass-spring system works, while Henning has no clue about environment mapping. However, we could effectively split the work.

A short version of our development schedule, with comments, is below.

20.3.2007 – 3.4.2007

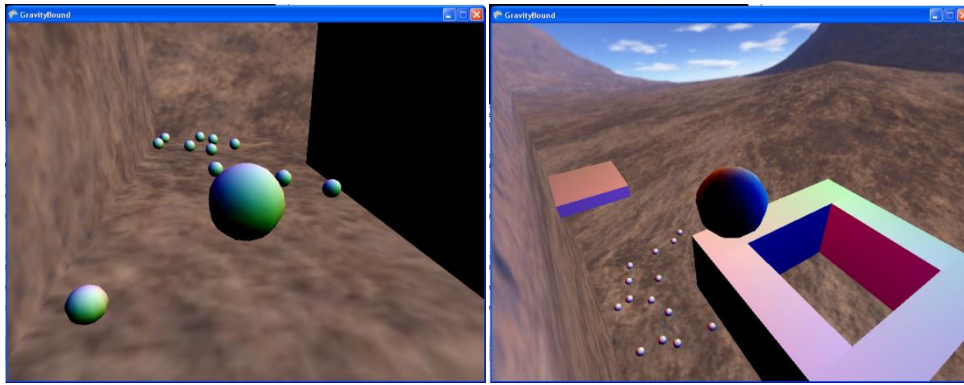
Project proposal

3.4.2007 – 17.4.2007

These two weeks we spent working into the XNA framework and starting with basic shaders, collision detection, blob physics (as rigid body), rendering of boxes and a skybox. The idea was to get a simple, yet playable version as soon as possible.

17.4.2007 – 24.4.2007

Within this time, we wanted to complete our functional minimum, including a camera following the blob, collision with internal game objects so you could jump around on them, you could die on monster collisions.



This is how the game looked back in the early stages. Very simple.

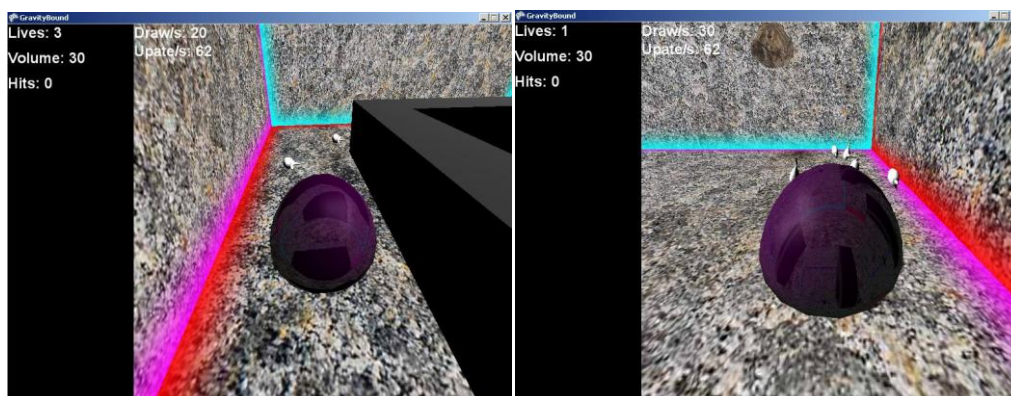
During this time, we were a bit ahead of the time schedule we initially planned. This was due to the fact that we worked quite a lot, but also due to the fact that some things turned out to be simpler than we thought. For example, the camera was a rather easy job, but we initially thought that it would probably take lots of time to get a good camera. This meant that we could start on other topics like the mass-spring system.

24.4.2007 – 15.5.2007

The goal was to reach the low target. Aspects like the gravity change and the mass-spring system were introduced and implemented. Sounds were integrated into the game.

The work on the mass-spring system was something that took more effort and fine-tuning than expected. Other things took more time as well, so we lost some of the advance we had with respect to the time schedule.

Here are two shots of the game on May 15th. One is just before, one just after a gravity change. Environment mapping can already be seen on the blob, even though it was planned for later.

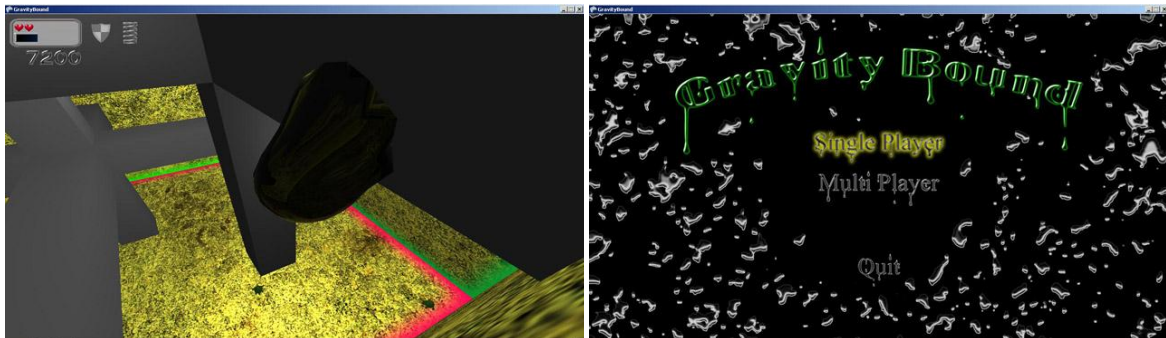


15.5.2007 – 5.6.2007

On June 5th, we had to have an alpha release version, so we had to invest quite some time to get things working. Multiplayer split screen was introduced, as well different AI for different monster types. Bonus items were implemented. The HUD was reworked and made better. We got rid of the black sidebar and instead had the information free-floating in the game screen.

The game formed and started to become a “real” game and feel like one.

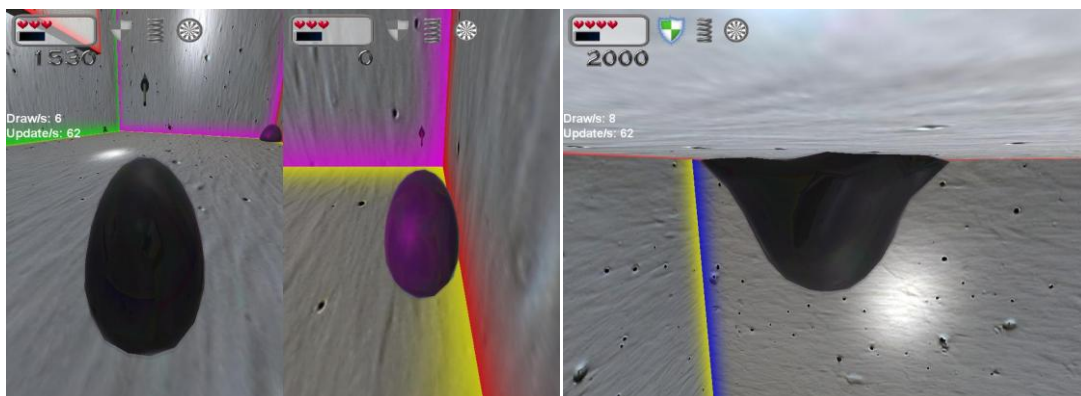
Lots of things happened in this time. We added a menu screen, introduced several levels, and scores. The blob started to lose and gain volume (and give visual clues about its volume) when shooting or recollecting old shots.



As you can see, the sticking feature is already active as well. The menu screen is in place. However, the internal level structures are still untextured.

5.6.2007 – 19.6.2007

Again, a lot of things were improved during this time. Internal texturing was added, the blob got more stable, play testing was done and lots of bugs were fixed. Last minute additions were done, more items went into the game and were implemented, more and more levels were invented and included. The changes were not too drastic, but after this, the game looked much more polished than before.



Play testing

After a long phase of development we introduced our game to commoners.

The whole play testing was done in a quite ad-hoc manner. We decided to get one or two testers for each mini test session. The reason behind this was the idea that the different testers should not influence each other.

We gathered 7 people in 5 different test sessions. Three of them where single player and two multiplayer scenarios.

The people were mostly computer science students we gathered at random; men and women. To further help us in understanding how we should interpret the playing behaviour of our testers; we asked some preliminary questions.

- What gaming experience do you have?
- What game control systems are you used to?

As the result of these questions we could separate the testers to three groups:

1. People having no gamepad experience. It turned out that asking them about their gaming history didn't make any difference at the end.
2. People with gamepad experience and an understanding of our type of control system (combination of semi separate camera and moving controls).
3. People, who had gamepad experience, but were not used to the input system we developed (the games they mostly played only used one steering cross with a controllable camera).

We have taken notes while the testers were playing, and tried to bring to narrate about their experience.

Given their impressions and the questions we asked ("What would you add/remove if you could?", "What did you like least/best?"):

- Some testers liked way it how it feels to play/be a blob. It felt realistic. They also liked its oily and shiny look.
- Most testers wanted to have better looking enemy monster and more graphical content.
- Some did not like the gravity feature, for they felt it was too complicated to use it.
- Some thought that the controls were too complicated.
- Most testers wanted some sort of first person view when you are in the sticking mode. It would make shooting much more fun, when hanging down from the ceiling.
- The whole setting seemed to be too dark.

Following these responses we tried to change the game accordingly to the critiques. For the improved graphical content we couldn't make any immediate changes, but we hope that over time we could produce more content. This in the end turned out quite well. We included new levels and more graphics for new enemy monsters. We also added a quite substantial amount of ambient light to make the overall experience brighter. We also added a first person camera to the third person camera to enable a first person shooting like game feeling. The cameras can be switched dynamically.

The other parts of the critiques (gravity change/control system) we left unchanged. The reasons for this decision were simple. The testers complaining about the controls came from the group, which had no experience with gamepads or were used to other control systems. So there was a natural obstacle the testers had to overcome. For the gravity change feature, which was essential to our game play and thus it was a bit sad that some testers didn't like it, we tried to produce level, where a combination of sticking and changing gravity would be much more appealing and necessary.

Conclusion

Overall, the course has been a great experience we consider our game mostly as a success. Mostly because we are missing artwork both on the sound and graphics part which would make the game much more pleasing.

On the technical side, we were disappointed by the .NET framework on the Xboxes well as the XNA framework itself. The most difficult part in our game was the mass-spring system for the blob itself. The massive use of floating point operations degraded the performance on the Xbox to an unacceptable level, so we started to concentrate on creating a PC game, rather than an Xbox game. The issues with the Xbox were^{1,2}:

1. Almost no operations are inlined, this includes Vector operations!
2. .NET framework on the Xbox is an embedded version, where some features are turned off
3. Tools available for memory monitoring and profiling were not sufficient
4. Non-generational garbage collector. System which create a lot of temporary objects can suffer in performance
5. AltiVec units for floating point arithmetic are not being used

All of these issues made us spend a lot of time looking at performance and doing micro-optimizations which all in all should not have been necessary. It is our opinion that this time could have been better spent on enhancing other parts of the game. This led us to the conclusion that it would have been better for us to create a PC-Game, possibly using C++, as there would have been a lot more resources and libraries available.

All in all, we managed to create a game that is – as is – a complete game, and at the same time fun to play. We were successful, and we are somewhat proud of that.

¹ <http://blogs.msdn.com/netcftteam/archive/2006/12/22/managed-code-performance-on-xbox-360-for-the-xna-framework-1-0.aspx>

² <http://blogs.msdn.com/netcftteam/archive/2006/12/22/managed-code-performance-on-xbox-360-for-xna-part-2-gc-and-tools.aspx>