Game Programming Laboratory '07

Parasite Paradise

Final Report

Daniel Wolfertshofer
Johanna Wolf

1. Introduction

The idea of this game is inspired by the PS2 game ‘Shadow of the Colossus’ where the player has to defeat colossal enemies by climbing onto them and stabbing their weak points with a dagger. The game level is not a mere static landscape anymore, but a moving creature with new challenges to the player.

Shadow of the Colossus

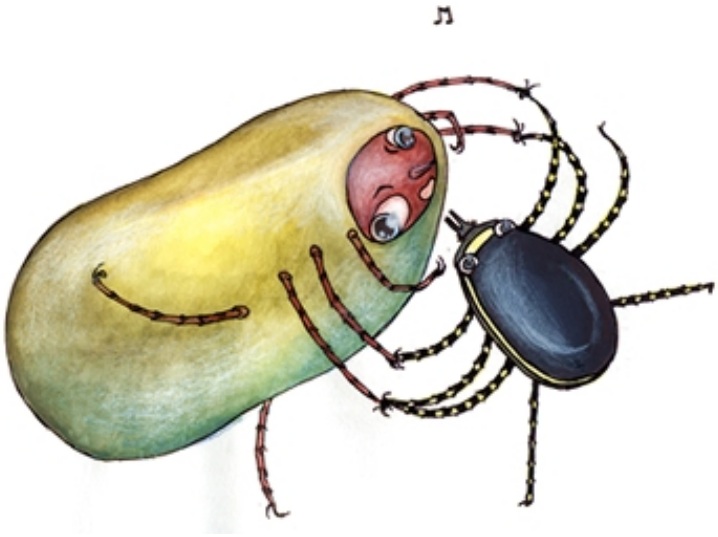
‘Parasite Paradise’, however, steps further into megalomania: The player controls a louse which crawls over the body of a huge moving fantasy creature. Each game level consists of one such victim. The louse starts off at a lower body part, a foot for example, and has to head for the upper parts. The goal is to reach the victim’s weak points where the louse can suck blood from until they have run dry. A victim will eventually fall unconscious from blood loss. Before the louse can move on to the next level, it takes advantage of its victim’s knock-out and lays some eggs. Until the breed has hatched, the louse has to protect them against a vicious hungry bug in an end boss fight. Finally, the player gains score points depending on the number of eggs hatched safely.

On its way to the top, the louse has to master some difficulties: It has to fight competing parasites by throwing eggs at them or making use of its claws like swords. When the player louse is attacked and injured itself, it weakens and eventually falls off the victim, which means that the level has to be started again from the beginning. When the louse sucks blood from a weak point, its energy is restored.

The louse has also the ability to jump, which has to be timed with care, however. Since it is sitting on a moving creature, a miscalculated jump can lead to a fall to the ground and a return to the level start. It is also not a good idea to jump when the louse is in an upside down position. As long as the louse is in contact with the creature surface, there is no danger of falling off.

Additionally, the louse has to beware of anti-lice shampoo floodings. The player has to anticipate the flow of the liquid and avoid it as it weakens the louse and could wash it off the creature’s body.

To make a different challenge of each game level, they contain special scenarios or tasks. For example, the victim could fly through the sky or dive under water which makes jumping impossible. Or the player louse has to win a race against a horde of enemy parasites and reach a blood-sucking spot faster. Eggs of vicious bugs could be hidden all over the level and have to be spotted and destroyed. The victim creature might also try actively to get rid of the louse by scratching itself, rolling on the ground or shaking itself.



2. Design

2.1 Assessment

The main point of 'Parasite Paradise' is the fact, that the level is moving. In contrast to a static landscape, the player is challenged by estimating the right time to jump depending on the movement of the ground he is standing on at the moment.

The game contains action, as the player has to defeat enemy parasites by throwing eggs at them. There are also adventure elements since the player has to search the creature for weak points he can suck blood from. Furthermore he has to complete tasks such as collecting hidden eggs or winning a race against other parasites.

The quirky game scenario provides a source of incorporating many surprise effects. Crawling over a huge body which is inhabited by a whole civilisation of parasites should give the player the chills and a unique game experience.

2.2 Technical Issues

During the game development, among others the following main techical issues will arise:

- Moving over an animated 3D model
- Collision detection
- Model animations
- Physical behaviour of moving, jumping and throwing eggs
- Rendering of viscous liquid using smoothed particle hydrodynamics
- Rendering hair by representing each strand as a B-spline for close hair and approximating it by sampling particles or sparse hair guides for distant hair
- Under water effects
- Rendering cloth for the level design

2.3 Look and Feel

The graphical design of the game is not aiming for persuasive realism as for example a human body for a level. Rather, the characters and backgrounds are intended to have a fantasy or comic style touch, looking like anthropomorphic toy insects. The look shall be reduced and colourful, possibly using the cell shading technique. Since the game outline has some quirky aspects, the design should be matched to this by aiming for a mixture of weird and cute.

2.4 Level Design

As for the levels, they have to be restricted to a very simplified design due to technical issues. The surface has to be smooth and limbs should be rather short. The underlying mesh has to be continuous, that is, there can't be any gaps, overlappings or intersections.

There doesn't have to be much focus on the background, since the moving level occludes it completely or the larger part of it mostly. It is a simply textured hemisphere and flat ground.

To attain the illusion of the level moving forth, non-static background elements could be added. One possibility would be tree models which move by or leaves falling in wind direction.

2.5 Animation

Each level has a small set of predefined movements. Within the background, the monster is not actually moving around. It will be at the origin and stay there from start until end. The default moving will be walking, but caused by certain events, other animations can be triggered. It might shake its body when the flea is sucking blood for example.

The character animations, that is the flea and the enemy parasites, consist of the following: walking, jumping, throwing eggs, tumbling and standing still.

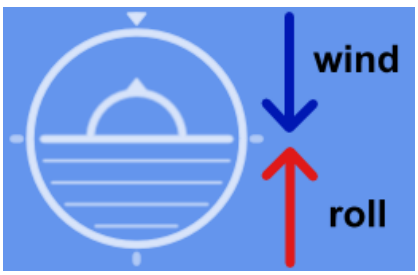
2.6 Camera

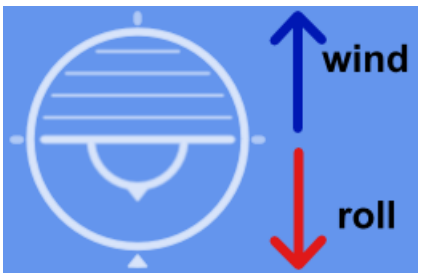
Having a static camera could make the flea disappear behind the level while moving over its surface. Therefore it is important that the camera follows the flea in some way. Since the level doesn't move away from the origin we could imagine the camera moving on or within a single hemisphere, always a fixed distance away from the flea. As long as the flea is visible from some point on the hemisphere this wouldn't be so diffucult. If this doesn't apply - because the flea is occluded by some part of the level - the camera has to be forced to move to another position in the hemisphere. These transitions need to be made smooth, so that the view is not flipping back and forth.

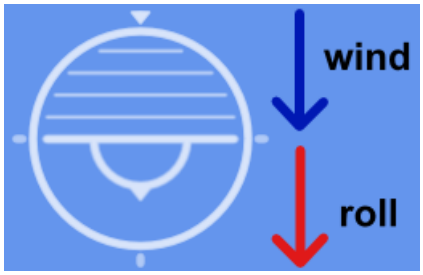
2.6 Game State Overlay

The game stats will be displayed by 2D sprite overlays referred to as HUD (head-up display) in the following. Besides visualizing properties like health or the amount of sucked blood, there is an artificial horizon like those used in aircraft altimeters. This should give the player an indication about the flea's spacial orientation to make a decision whether it is a good idea to jump at the moment as the flea might cling to the level upside down and fall off.
The HUD prototype occupies the upper left and right corner of the game screen. The design idea is that the left part indicates general physical facts important for movement while the part to the right displays gameplay relevant details.
The left HUD shows an artificial horizon giving a simplified impression of the flea's roll (in flight navigational sense) and therefore indicates to what degree it is upside down. Around the artificial horizon an anemoscope or weather vane is drawn also restricted to the plane where the roll of the flea happens. Together they give a quick overview of the current situation when it comes to judging whether or not it is a good idea to jump right now.

The right HUD shows both the flea's and the level's health status as diminishing curved bars. The health bars surround a simple radar view giving a 2D impression of the distance to the nearest opponents which are shown as dots. The middle of this radar views circle is the fleas position in the displayed plane.

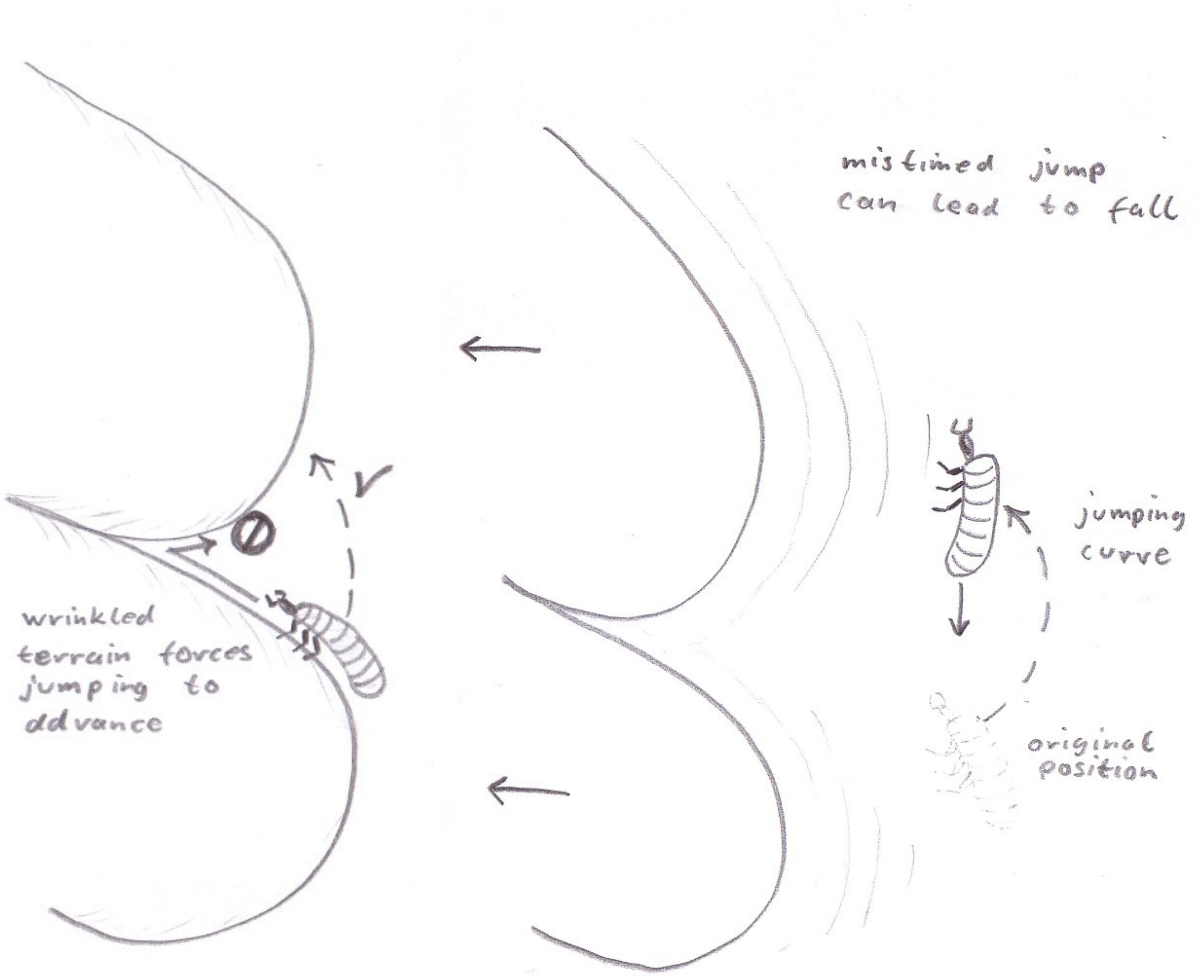






2.7 Jumping

When the player triggers a jump he can influence the trajectory by pressing the thumbstick in the desired direction. While flying through the air, there is the danger that the flea will fall off the level. For example, the body part the player aimed to jump on has moved away meanwhile or the flea had been hanging upside down from the body. As the danger of falling down could prove as a source of frustration we are thinking of ways to giving the player more control over jumping if playtesting will suggest it. An idea would be to give the player a second chance when he succeeds at a reaction test shortly after the fall has occurred.



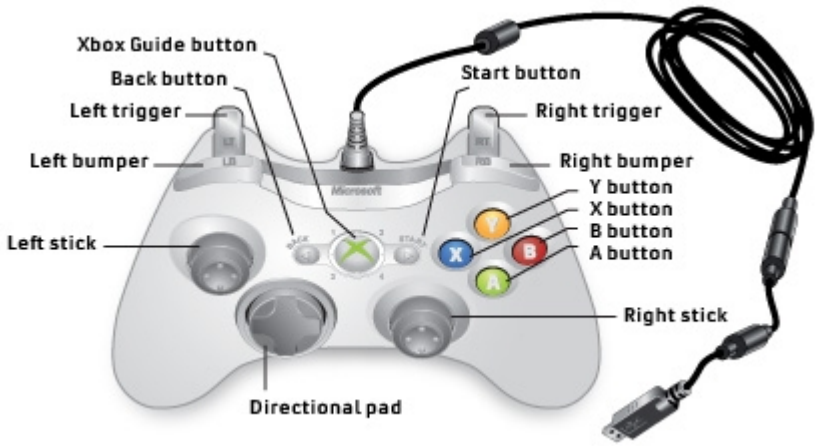
2.8 Fighting

There are currently two ways we envision the flea to fight its opponents: Firstly, the opponents can shoot each other by throwing eggs. The shooting direction is fixed to the character's heading forward direction. The flight width of the egg can be steered however by pushing the trigger button at different strengths. This is only possible with the Xbox joypad.

2.9 Controls

2.9.1 Xbox

- Left Stick: moving, jumping direction
- Right Stick: camera
- Right Trigger: jump
- Left Trigger: shoot
- A Button: suck blood
- Right Stick (pressed): zoom in/out
- Start Button: pause



2.9.2 PC

- W/S/A/D: The louse can move forward (W) and backward (S) and turn left (A) and rigtht (D)
- Up/Down/Right/Left: camera
- Space: Jump
- ALT: Shoot
- Q: Suck blood
- Z: zoom in
- U: zoom out
- F1: pause

3. Development

3.1 Task Breakdown

1. Functional Minimum
    - The player can move the flea (no jumping, no animation)
    - Static level with different ground properties (slippery, sticky)
    - Skybox
    - Overlays
    - Blood can be sucked from weak points
2. Low Target
    - The camera does not lose track of the flea
    - The flea can jump
    - Throwing eggs
    - Enemies with simple AI
    - Flea / Enemy health indication and management
3. Desirable Target
    - Collision detection
    - The level is moving
    - End-boss fights
    - Cell shading / Overall nice graphics
    - Sounds
    - The camera does not lose track of the levels movement
4. High Target
    - Hair simulation (no interaction)
    - Different game scenarios
5. Extras
    - Anti-lice shampoo
    - Multi-player option
    - Cloth simulation
    - Collision detection with hair

3.2 Software Tools

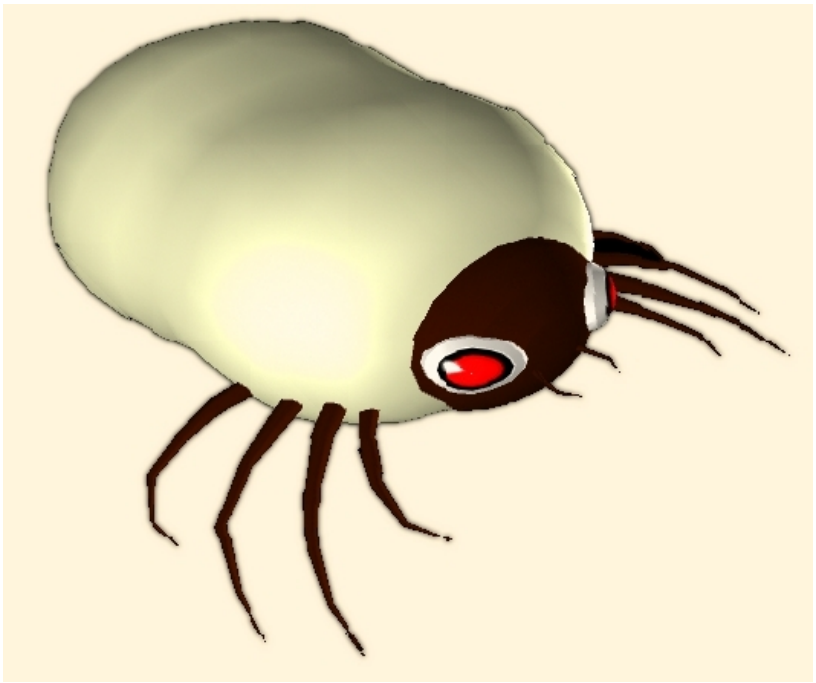
- Autodesk Maya: Modelling, animation
- The Gimp: Texture editing
- Sony SoundForge: Sound
- Sony Vegas: Video editing
- TortoiseSVN: Repository

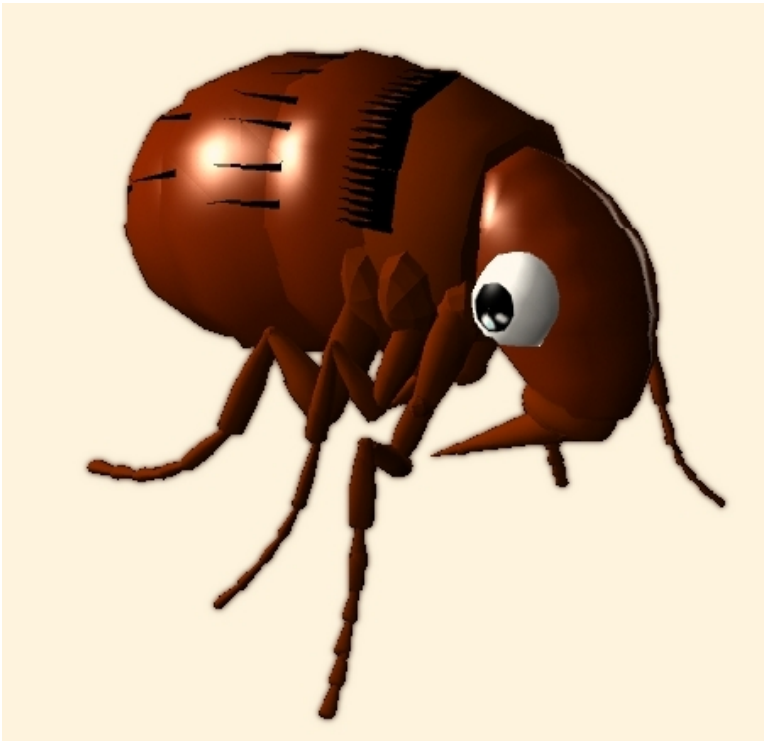
3.3 Final Project State

The functional minimum and low target have been fullfilled and part of the desirable target has been reached.

3.3.1 Achieved tasks

- Flea control: Moving around, jumping, shooting
- HUD: Display of the character health, the amount of sucked blood and the spacial position. Indication of nearby enemies.
- Camera: Follows smoothly the flea, can be controlled by the user, zoom
- 3D Models: Static skybox, eggs, animated flea, tick and bear level.
- Ground properties: the flea moves slower over sticky ground and can suck blood at certain spots.
- Throwing eggs and simple handling of hits on NPCs and obstacles.
- Particle sprites: Exploding eggs
- Very simple NPC AI.
- Collision detection: Bounding sphere collision detection between characters and eggs. Tree based hierachical collsion model for the level: currently only mesh vs. sphere shape is implemented.
- Navigation map: Moving over the level surface by tracking the mesh triangle, a character is located on. When it crosses a triangle edge, the next triangle is assigned and the character's local coordinate system transformed properly such that it stays on the surface.
- Animation: Controller to play arbitrary keyframe ranges.
- Sound: Background track taken from the SNES game 'Earthworm Jim' and bear growling sound effect.





3.3.2 Technical difficulties

- Import of animated models from Maya: Our first approach was to convert the fbx file into an x file and use a very convenient animation library from the web, but we had to find out that there is no x-file export plug-in available for Maya 8. Then, we decided to use the skinning sample as mentioned above. This meant however, that we had to implement the animation controls and do the shading via the effect file. As we had to focus on other issues later on, we gave up on the shading effects and turned to texturing the models.
- Camera: We first used a convenient camera implementation from the web, HMEngine, which however turned out to be partly incorrect. So we decided to throw it out and implemented an own camera which met our simple requirements.
- Collision detection on an animated level: The navigation map works fine on a static model. The step to an animated level would only involve applying the proper skinning transform matrices to the vertices. Still, retrieving the correct indices from the vertex buffer proved to be a major obstacle and remains unsolved.

4. Conclusion

We believe our game idea was a rather a unconventional and interesting approach that imposed complex and challenging tasks to achieve. We had many ideas we would have really liked to implement, such as the hair or anti-lice shampoo simulation. Sadly our vision contrasts reality as we had far too less time to achieve our goals. Already the most fundamental element – triangle-based collision detection – turned out to be a hard to overcome obstacle. Being stuck with details, we had no chance to focus on the actual game functionality.

While we pity very much not having been able to realize more of our ideas, we do not think the time we spent on the project has been wasted. We learned a lot using various tools, especially Maya, and most importantly had fun while developing and designing.

The course itself was quite well planned considering that it has been offered for the first time. There are only a few minor points where we would like to suggest a reconsideration of the course schedule:

First of all we think one of the very first lessons/labs should be an in-depth tutorial on how to deal with XNA specific issues most notable the content pipeline. In our opinion the content pipeline is one of the things most teams had problems with. A tutorial on how to import custom content and model details could help a lot.

Another thing that might be worth considering is whether some assignments could result in a less formal presentation. For example instead of having each group present their personal feelings about XNA why not have them just sit together and chat about it. We could imagine that this might produce more usable insights than a more formal talk.

In general it might be a good idea to use some time of the lessons/labs to just chat about common arising problems between all course participants. While one often meets other groups in the computer rooms the oportunity to have all together seems natural to exploit in this way.