Developing a Jump'n'Run Puzzle Game

Results of a practical course at the Chair for Computer Graphics and Multimedia
(RWTH Aachen University, Germany)

Paul Varney Sven Horn

Abstract

Princess S. Cape is a JumpnRun Puzzle game. After various heroes have failed saving the princess she gets bored and tries to escape on her own. To do that she has to find her way through a labyrinth of rooms, find keys and avoid traps.

Keywords: game programming, jump'n'run puzzle game

1 Gameplay & Features

The idea for the gameplay was based on the Game Continuity. [con] It is basically a JumpnRun game where you can (and have to) move around the rooms of the level to proceed from one room to another (see Figure 1).
There are various assets in the game the player interacts with. Keys can be picked up to activate doors; there are moving platforms, traps, ladders and jumpboxes. After having completed a full level and finding the exit the whole level set flips around and another world starts on the backside.
To make the world a little more vivid there are small animations like animated torches on the walls or clouds and balloons flying by the windows.
The graphical style of the game consists of cartoon-like textures and hand-drawn assets. The target was to make it look like a stage design from a theatre play or like a dollhouse. The background assets are 2 dimensional and sometimes that can be seen when running around in the levels.
We tried to realize this by having different layers: The first one would be the layer in the front with the player figure, interactive objects and non-interactive obstacles (like e.g. boxes). The second layer has the background objects that stand in the middle of the room. Then there is the back-wall layer, which is basically a wall with some decorations and some windows. Through these windows the background/horizon layer can be seen. On this layer there are also moving objects like clouds and/or balloons.
Having the objects arranged in this layer structure gives the feeling that the boxes and bookshelves are not real but only there to make it more beautiful (like in a stage play as mentioned before).
The inspiration for this idea came from games like Little Big Planet. [lit]

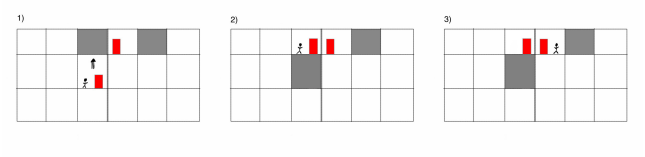

Figure 1: *Moving tiles to progress from one room to another*

Figure 2: *Implemented Metaview to move tiles*

2 Development Process

At first we started out with 5 team members of which, up to today, only two are left. This caused various problems and reduced the features we had planned in the proposal from October.
When we started we had several meetings and also some brainstorming-sessions via skype and teamspeak to decide the design, gameplay, features, game name and background story for our proposal.
We planned to do 4 levelsets on a cube with the menu on the top side and inventory on the bottom (see Figure 3). The idea of the

menu vanished quickly, since we had so much to do to get the game itself running we didnt gave the menu a high priority.
Right after we handed in the proposal two of our team members quit the course, leaving us with a five man proposal and only three students. After talking to our supervisor we decided to cut down the proposal a little and continue with the game development.
For the first milestone we had just one hardcoded room, our player figure dummy could jump and walk around. We used a simple boundary box collision detection. This is no problem because we do not use any other models than boxes. Here we just calculate if the distance between the player and the object is bigger than the addition of the player size and the size of the object. The problem was that we had a very low frame rate (40fps-60fps) caused by our badly organized code and by the way we rendered the scene.
Another problem we encountered while programming the game logics was that sometimes the player figure got stuck in the ground or even fell through it. This was due to the fact that the player figure started out too close to the ground and was pulled downwards by the gravity. When there was a little lag in gameplay it pulled the figure through the floor. We managed to solve that problem during the second milestone by decreasing the gravity value by factor 20 and only increasing it, when there no floor below the player figure.
One of us also worked in blender to make new 3D objects which, in the end, we didnt really need, because of time issues. The main reason for that was, that there were lots of other things to do and that the group together with our supervisor came to the conclusion, that other 3D shapes than cubes and simple faces would interfere with the graphical style we had intended, and that we would fake different shapes with textures. At this moment we realized, that we would not need an inventory, because the only collectable objects in the game were keys. We decided to rather have them open doors or activate other movable objects directly when picking up the key.

In the second milestone we almost had a fully working game. Most of our interactive assets (like ladders, keys, moving boxes and jumpboxes) were working as intended, the metaview worked correctly and the flipping of the world after a level was completed. This was also when we had the decision to have all levels as indoor levels and keep the graphical dungeon style as the main style.
For the third milestone meeting we had our code reorganized and cleaned up which improved the framerate drastically. It also made working in the code much easier because we had lots of different small classes for everything. We also finished the game logics (fixing bugs) in this milestone.
The only thing missing for the final release were some background assets to make the levels more vivid and the graphical effects one of our teammembers was working on.
We then added the background assets and added a few animations (like the walking animation from the player figure).
During all this time our long time third team member was working on graphical effects. He implemented Depth of Field, which is also used in the final version of the game. He also worked on shadow mapping and other effects or fixes of graphical glitches like the tearing textures due to wrong scaling. Sadly, on the day of the final deadline he told us that he had left the group and quit the course with all his work packages unfinished and no chance for us to finish them. This led to problems and to a game without the intended graphical effects. If we would have known this earlier, we could have tried to complete the features he was working on by ourselves. This was caused by communication errors, but it was hard to manage because he always told the rest of the group that everything will be finished right on time.
Maybe redmine could have helped to track the status, but if someone like in this case would fill in wrong information, even redmine could not have prevented that from happening.

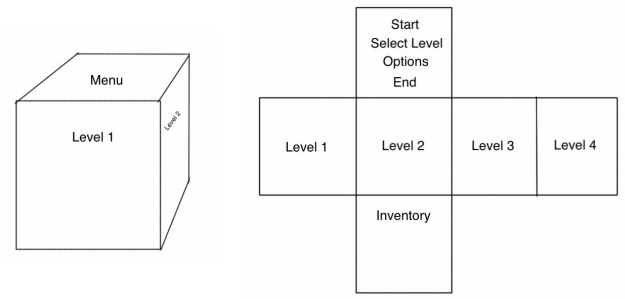

Figure 3: *originally planned cube layout*

3 Conclusion

After all, it can be said, that we have planned much more than we realized in the end. The main reason aside from the reduced team size was that we did not have much programming experience before we started this course and underestimated the workload our ideas caused. We also had communication issues in the group like e.g. we did not use redmine enough to track our progress, so we sometimes lost the overview over the things that had to be done or fixed. We have learned a lot in this course. Our programming skills advanced and we finally got the impression how much work even such a little project causes in addition to the programming.
For versioning we would have preferred using a tool with a graphical interface, since it's just easier to use than command line tools. Git did work fine, but we needed some time to figure out how it works, and sometimes still mixed up some of the commands.
For future projects we would definitely use redmine more to make tasks clearer and working more organized.

References

Continuity. http://continuitygame.com.

Little big planet. http://www.littlebigplanet.com/.