Te3Ds – Final writeup

Bipeen Acharya, Olivier Mahame, and Suraj Bajracharya
Computer Science Department
Washington and Lee University
Lexington, VA 24450 USA
Email: {acharyab15, mahameo14, bajracharyas14}@mail.wlu.edu

Product

As our final project for Integrated Computer Graphics, we thought of a re-imagination of the classic game of Tetris which was to become Te3Ds. Based on our initial vision of the project, we believe we surpassed our expectations. Initially, we had thought of designing a game very similar to Greg Zipkin's Tetris3D game in OpenGL. However, as we started experimenting with different features available to us in webGL, we started finding our own ways. And we designed a game which is very similar in the functionalities, but also very different – we have even integrated the remix of the classic Tetris soundtrack as the soundtrack of the game. The game in itself might face difficulties in getting people to keep playing, because it takes a long time for anything exciting to happen in the game as the container is too big.

Interface

We have a fairly efficient interface that allows good user interaction with the game. We have implemented all of the geometric transformations we have discussed – Scaling, Rotation and Translation- that allow users to interact with the Te3Ds components. The camera movements are also very efficiently integrated, albeit with a few bugs here and there.

Differences from initial vision

Talking about what we did not do as initially thought of, lighting is probably the biggest component that we did not use. When we thought about the game, we thought of shadows and thought of implementing lighting. But we realized that it would be much simpler just using projections to project the area where the Te3Ds component is falling. Therefore, we removed lighting from our project completely, which also made things much simpler. Secondly, we promised to implement "Texture mapping (if time permits)" but we came to a decision not to do it, because of time constraints that occurred due to some unforeseen difficulties that came up. Suraj also said that he didn't think it was necessary. We also thought we would be using blender to render objects into the canvas. However, we decided to render cubes and use those cubes in making the Te3Ds shapes by putting them all in the right positions.

Difficulties and problem solving

Since we started with our CAD program in our webGL project, the start of the project was pretty straightforward. The mouse rotations and zooming in and out of the game, we had already written codes for. The design of the canvas, with the scores and level information, was fairly straightforward too. Although we did have to spend some time on figuring out how to link the object that is coming next to change the image that we display of the next shape in the game.

To take the other road, we came across some serious difficulties when developing this game. We said using projections was much simpler, but only after we had spent hours being stuck trying to figure it out. To do the projections, we check x and y values of an object and compare with x and y values to all the previous objects, calculating the maximum z of all the x and y values. This was really counter intuitive for us but we decided to go with it, since we did not have a better idea.

Object rotations and object collisions were also fairly difficult to get a hang with. We had a lot of bugs that we constantly came across – one even involved the shapes flying into space. We used the small cubes to make the shapes instead of using a single object because we thought that removing a layer after a player filled one would be easy. However, that decision made rotations of the shapes very difficult. All the smaller individual cubes rotated differently when we implemented rotations, giving us faulty images and rotations. So now, we have calculated the distance from all the small cubes to the origin, and stored it in an array

and taken the minimum distance to determine the required transformations that are applied to all of the small cubes.

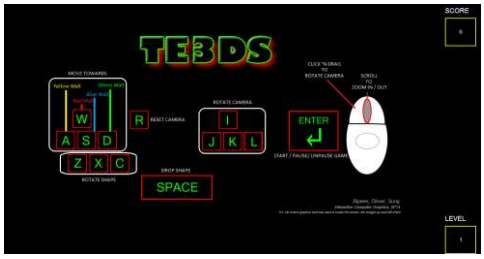
Another difficulty we faced was in restructuring our code every time we needed to add a new feature. For example in order to implement the filled plane removal we had to create 3 different object tables (grid table, projection table, and cubes table). This way we were able to manipulate different type of objects independently and re-buffer data to GPU with ease.

Conclusion

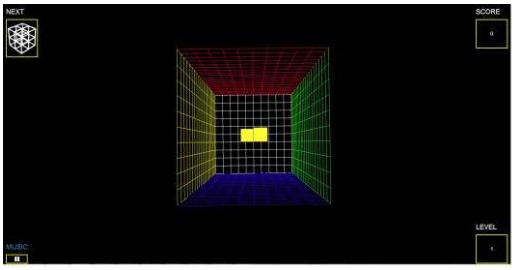
Overall, this was a very interesting project to work on. We learnt a few very important lessons, about what we should and should not in terms of writing and maintaining codes. We also got to learn about the Javascript API and webGL at the same time. We definitely think we did a very good job with the project, and maybe if we get positive responses from people (which we have already started getting), we might even keep working on it to make it better.

Song Reference

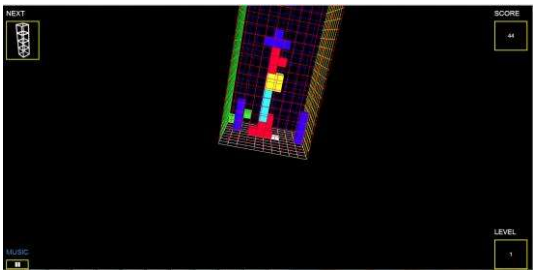
Song Reference: London Philharmonic Orchestra; Skeet A. "Tetris Theme (Korobeinikj)." Korobeinikj. The Greatest Video Game Music (Bonus Track Edition, 2011, iTunes.



Game instructions



Game default view



Rotation of container