Climbing the Hill

A Computer Graphics Project in the Liberal Arts

Alex Rosenthal, Bryan Vaihinger, Tinotenda Zinyama and Elodie Fourquet
Department of Computer Science
Colgate University

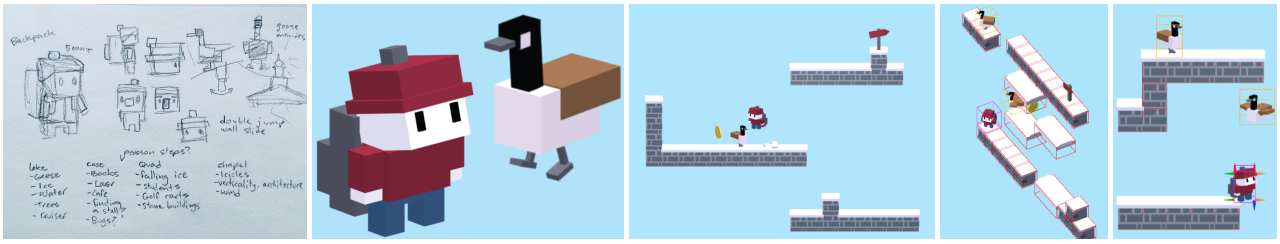

Figure 1: The main characters were first drawn on paper. The simple abstractions are turned into a combination of 16 boxes for the main hero and 10 for the goose. The level 2 platform shows the main character jumping over the goose. An angled view of a platform scene, demonstrating the first attempt at bounding-boxes based collision. The refined geometry of the main character to tweak collision detection.

ABSTRACT

This poster presents a student final project for an Introduction to Computer Graphics course at the undergraduate level. Taught in a liberal arts setting, students' interests and multiple expertises helped the visual quality of the 3D game produced. In particular, the students allied their practices of art and science with their technical ability in programming to craft an engaging platformer game for the web. The challenge given the time frame and the student's ambition was addressed by committing early to a simple design and by personally implementing the basic physics needed, rather than relying on external assets and extensive libraries.

ACM Reference Format:
Alex Rosenthal, Bryan Vaihinger, Tinotenda Zinyama and Elodie Fourquet. 2018. Climbing the Hill: A Computer Graphics Project in the Liberal Arts. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

CONTEXT

In a liberal arts institution students are free and encouraged to explore a wide range of academic subjects. Students may choose to study Computer Science alongside English, Economics, Chinese, Theatre, or Fine Arts for example. To permit such varied combinations of expertise, the Computer Science major only requires students to complete eight courses at our institution. Nevertheless, our students' strength lies in their broad interests and varied expertise, as revealed by the open-ended project presented in this poster.

Given the liberal arts setting, our computer graphics course emphasizes both the technical and artistic sides of the discipline. Attending guest art lectures and gallery openings, practicing critiques of graphics students' work enriched the classroom's appreciation and knowledge of the art and science required to produce and manipulate effective CG images.

For the final course project, students proposed a graphics application of their own design in small groups. The goal was to integrate the practice of the fundamentals of computer graphics (using WebGL—the JavaScript API based on OpenGL ES 2.0—and `three.js`) within a theme of their choice. In four weeks, dedicating (about 50 hours development time), the student authors wrote a game designed with simple primitives, directly composed and animated in code so that the main challenge is to complete fun and increasingly difficult levels. Each level is made of static and moving platforms, with coints to collect and geese to avoid.

1 DESIGN

The first step for designing the game theme was to draw on paper. The left image of Figure 1 is part of this planning stage. With sketches students conceived the visual look and narrative of the gameplay elements, inspired by the natural setting of their campus up the hill.

Taking into consideration the time limit and the importance of visual coherence to producing a complete game, simple approaches and abstract modelling were prioritized from the ground up. Thus, the models were created entirely in code, using a handful of 3D primitives for each element. Similarly, the camera view was constrained to a front orthogonal projection and the gameplay is based on a 2D logic.

In addition, animated parts were given simple motions. The coins spin (as per usual in games), while Newtonian physics controls velocity for falling objects (Figure 2). The main animations are thus governed by simple mathematical functions, and given subtle behaviors such as blinking to give the illusion of life. Students spent time tweaking the playable character to give them movement lead-in, variable jump height, and other detailed animations.

Conference'17, July 2017, Washington, DC, USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

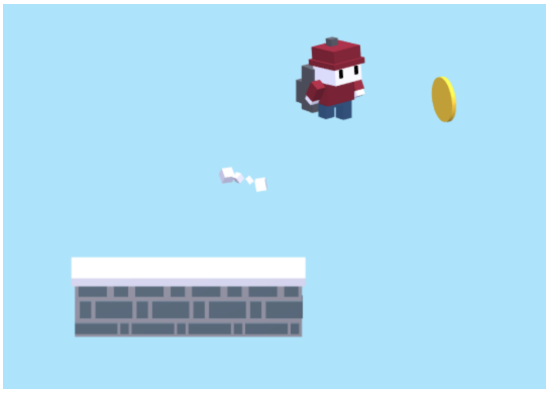

Figure 2: Main character execute a double jump. To accentuate the affordance, since an important gameplay twist, a small particle effect is displayed.

The conservative scope of the game's design and assets provided time for students to polish the material they created. Because they coded everything from scratch, students knew how to customize each feature. They maintained full control over the gameplay, appearance, and interaction.

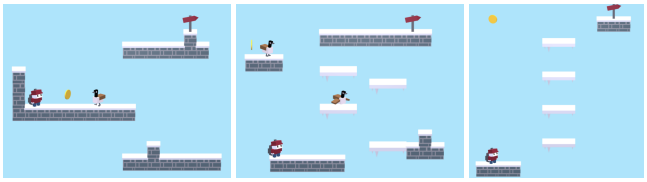

Figure 3: Sample levels showing the difficulty progression of the game.

One drawback of this project structure was that some objectives weren't achieved by the final build deadline. The original proposal envisioned several "zones" with their own assets and gameplay elements: such as a library level, or reaching the top of the Chapel (a landmark of Colgate).

Nevertheless, as the video demonstrates they created a sleek, attractive and cohesive game. The 8 levels gradually introduce new pieces and twists to build a challenging but understandable journey. Figure 3 shows how the levels become more complicated, while retaining a vertical structure to evoke climbing Colgate's campus hill.

2 DEVELOPMENT

Developing the game successfully as a team in four weeks required to establish, communicate and synchronize the shared code with an effective separation of concerns. Students choose to split responsibilities: one member prioritized modeling and animation, another focused on game logic and collision detection and the third developed a text-based level generator. Each student incrementally developed their set of features, starting from a basic implementation and adding scale and complexity as mechanisms were tested and integrated in the common framework.

Figure 4: A string encodes a level layout.

For example, the level generator based on a grid layout string encoded first the main character and the ground. As shown in Figure 4 it was later extended to a more exhaustive scene layout and interactive elements. Each level element is then created and placed based on the grid-based encoding character. Elements share a common interface, but moving elements have added behaviours that are not tied to the grid-level editor.

Another challenge that was approached iteratively was the collision mechanism. The use of a dedicated library, such as `Physi.js`, was rejected early on for being overblown given the physics problem at hand. The first approach of using axis-aligned bounding boxes (as shown in fourth image of Figure 1), was however deemed insufficient. The playability of the game was greatly enhanced by augmenting and tuning the approach to correspond to the main visual contact and positive space in the character bounding box. By testing, an ad-hoc refinement that does not include the character's backpack was revealed to be the most effective game play.

Click to Play Online

Click for Gallery and Project code