Project Proposal: Raider Run

Ron, Chien, Laura, and Shoshi

Statement

Our goal is to develop an interactive 3D game in which the Colgate raider evades objects falling from a tall tree atop the Colgate Hill. The user controls the raider with arrow keys to move around the tree to avoid the falling objects. As the player progresses, the game increases in difficulty as the obstacles fall at a faster pace following each level. Our scene is atop a cartoonish minimalist version of the Colgate Hill. Our hill will be among the clouds as they are randomly generated in the background. The main focus of our project is developing a visually appealing yet fun game which is accomplished through fluid hero animations and cohesive graphical elements (textures/color scheme/collisions/etc). The game should also have a simple menu to help the user easily pause, change the difficulty, restart, and quit. While there is no way to "win" the game, the objective is to survive as long as possible.

Good points →

for evaluation and completeness purposes: There should be a winning condition.

yes! useful to debug All projects should have it. Thank you for reminding me

Technical Outline

Tree

The tree itself will tower over the hero, as most of the tree will be out of frame other than the trunk/base. The object itself will likely be a long cylinder which will be made to look like a tree using bump mapping and effective textures. The top of the tree (leaves) will be built using a simple green box design such as the example developed by leomanlapera[8].

You had a good point such look isn't well integrated with a low poly model

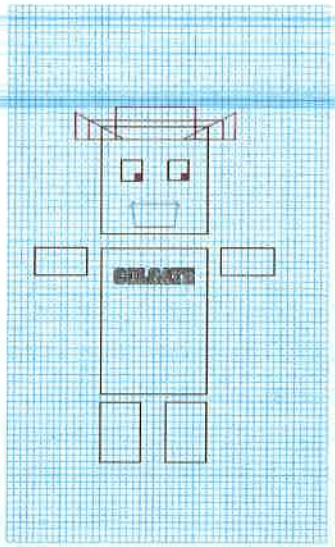
Hero

Our hero, the Colgate Raider, will have a similar look to characters from the game *Minecraft*. Each shape in this sketch will be a 3D version. If time, hair will also be added.

→ your hat works well right now

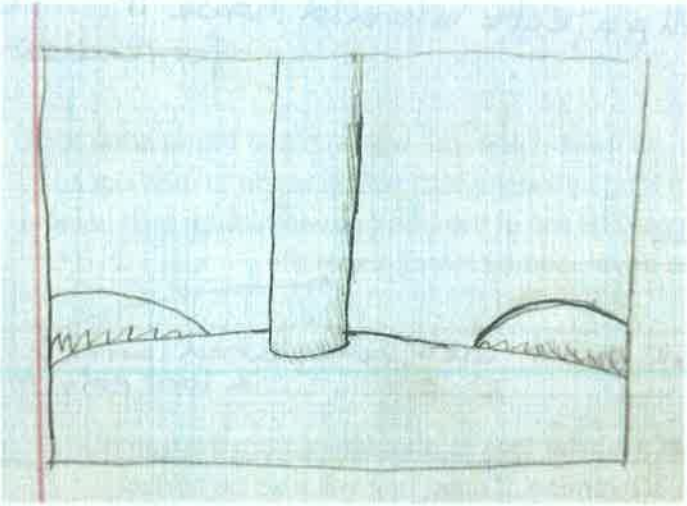
Have you thought about having many trees? that would introduce more opportunities to have the raider steer away from falling 'things'

You are in the right direction thinking about a 2D logic within a 3D appearance to focus on the fun and the visual appearance of your game.



2. Static scene

The static scene includes a foreground of a cylindrical tree trunk on a hill (modeled as a large sphere) from which objects will be falling. Behind this hill, there will be a few more hills, also modeled as spheres. Below is a simple sketch of what this would look like.



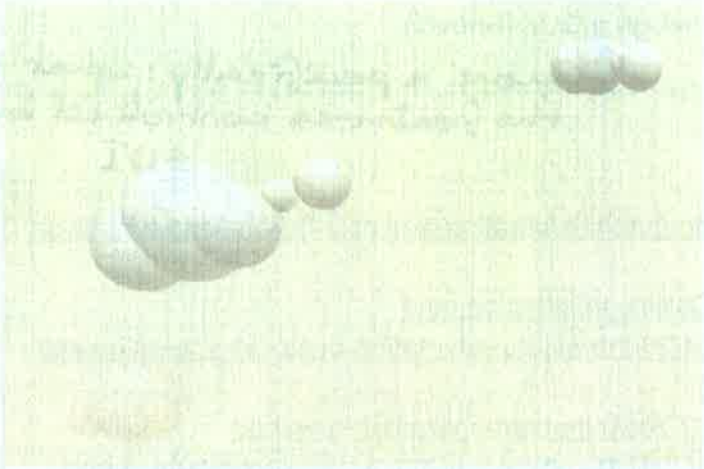
3. Hero Collisions

The game ends when the hero is hit, or collides, with an object falling from above. In order to recognize that a collision has occurred between an object and the hero, we looked at the collision detection function in the simple 3D endless runner game,[1] as well as another more basic collision detection tutorial.[2] In the endless runner game, the collision detection function calculates the distance between the runner and the obstacles and triggers a collision if they are very close to each other.

Good you looked at the code. Bounding boxes are useful to compute intersections between moving/scene elements static

4. Background

The background is composed of a sky with randomly generated clouds. For clouds, we consulted an Aviator game tutorial by Karim Maaloul [3], but we will make the clouds spherical instead of cubical to match the look and feel of the rest of the game we want to achieve, as shown below. This tutorial uses randomness in the number, position, rotation, and size of spheres.



Need to more different.
The goal is to be your own game: as you said 'main focus' visually appealing & fun.
(Chien already did this change, in probably few minutes)

6. Falling Objects Motion

Do you have ideas, or have you consider different objects?

The falling objects will be moving down towards the ground, and as the game progresses they will fall at a faster rate. This motion could be achieved with constant motion, or if time allows we could incorporate physics and simulate gravity to make the falling motions look more realistic. We could use Physijs, a physics plugin for three.js.[4]

Useful for collision too.

7. Character Movement

Have you look at the 'walking' 'moving' animation of Yakudoo

Our main character, the Colgate Raider, will be controlled to move left and right by the left and right arrow. This should not be a challenge until we add walking animations to the character. When the Raider moves, there will be a walking animation instead of the character model just sliding across the screen. There is a lot that goes into a human figure walking animation. We will look at two tutorials. The first is a *Minecraft*-style walking animation [5] by Alexandra Etienne and Jerome Etienne. The second [6] is by Karim Maaloul, who created the Aviator tutorial, and explored the animation of the Moments of Happiness demos[7] we looked at in class (What a coincidence; we found this on the second page of our Google search). What we would use from this tutorial are the sections on using trigonometry for walking or running cycles.

Objectives

1. Hero to be modeled after the Colgate Raider. (Shoshi)

2. Static scene with a tree trunk on a hill in the foreground and hills in the back. (Chien)
3. Recognize collisions when hero hits an object and loses the game. (Laura)
4. Background to include a sky with randomly generated clouds. (Chien) ← invalid objective as such
5. Textures will be applied to tree and hero to make them look more realistic. (Ron) → where? for the shirt?
6. Motions for objects falling from the tree. (Laura) ← what are the objects?
7. Animations for main character moving. (Chien) ← Do you have a tutorial to help you with this objective?
8. Animation for when object collides with hero. (Ron)
9. Algorithm for semi-randomly dropping objects. (Ron)
10. A game menu will be implemented through a GUI. (Shoshi)

↑ more specifically: what are the features controlled by the GUI

Sources

1. https://gamedevelopment.tutsplus.com/tutorials/creating-a-simple-3d-endless-runner-game-using-three-js--cms-29157
2. https://stemkoski.github.io/Three.js/Collision-Detection.html
3. https://tympanus.net/codrops/2016/04/26/the-aviator-animating-basic-3d-scene-threejs/
4. http://chandlerprall.github.io/Physijs/
5. http://learningthreejs.com/blog/2012/07/05/minecraft-character-in-webgl/
6. https://www.smashingmagazine.com/2017/09/animation-interaction-techniques-webgl/
7. https://moments.epic.net/
8. https://codepen.io/leomanlapera/pen/EWBZLW

Dear Ron, Chien, Laura and Shoshi,

You made some progress in defining your ideas, but you have to improve your design to come up with specific and sufficiently ambitious objectives.

The Raider sketch helps a lot; your commitment to the Minecraft looks frames the design you draw and will guide the implementation of this objective. Similarly, you have to determine a plan of action for its motion.

For most of rest the planning and description (objectives and technical paragraphs) is still too undetermine and vague. It seems you have decided to remove / stay away from rotation and that might be a good decision. We need to talk more so you extend and revise this proposal.

It can be a visually pleasing and fun game but you need to plan it better, while committing to features.

Best,
Elodie

Emrys Sledding Down the Ski Hill
COSC435 Final Project Proposal
Group Members: Asad Jamil, Zoila Rodriguez, Leslie Subaldo, and Jingxian Wu

Introductory Statement

have you thought about rocks?

Our goal is to design and implement a game in which the user is able to control Emrys, President Casey's dog, on a sled down the ski hill for a certain amount of time. The objective is to collect food tokens while avoiding obstacles such as branches. The game ends when the player collides with a certain number of obstacles or when time runs out.

low-level poly semi-sphere (terrible drawing, I apologize)

↳ then highest score board, or is it the winning condition.

The scene is able to be viewed from both first-person and third-person perspectives. When the game starts, Emrys will start in the center and immediately start sledding down the snowy ski hill. The trees are going to be on both sides of the ski hill, creating a boundary for Emrys' movement. In front of Emrys will be the various tokens that the user can collect and the branches that the user can avoid. These objects will be randomly spread out. Buildings of campus, e.g. Huntington Gym, will be rendered at the "end" of the hill to show campus in the distance.

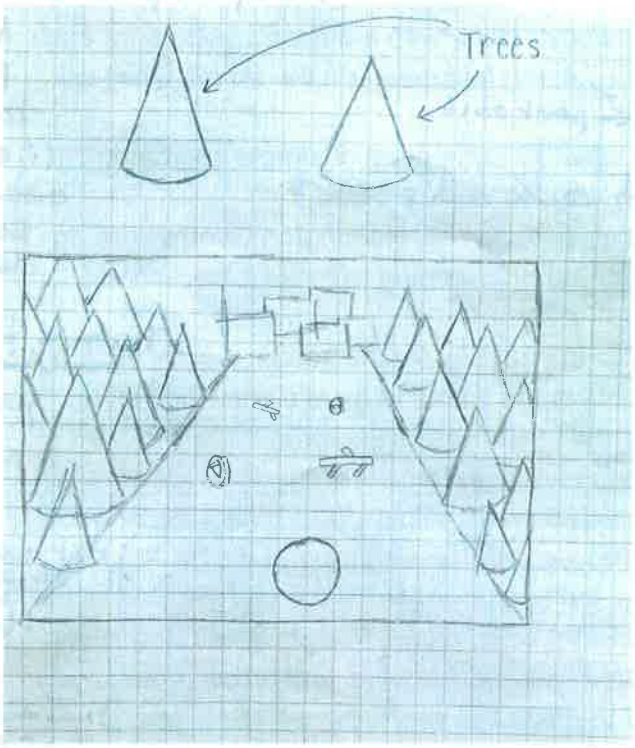
← maybe

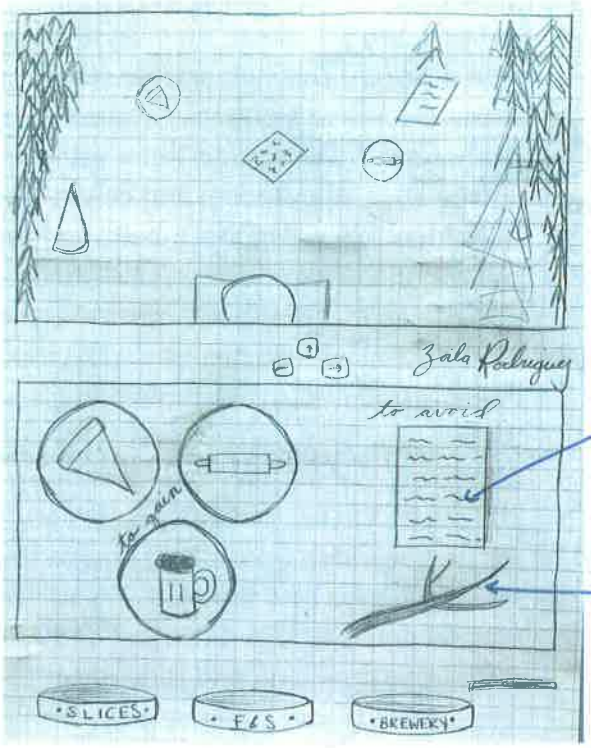
Have you thought about colors? too much white, light conditions/environment may be difficult.

The key user action in this game is to control Emrys as he sleds down the hill. The user can use the left and right arrow keys to move him left and right, the up and down arrow keys to speed up and slow down the sled speed, and the spacebar to jump over obstacles.

I have a suggestion: please talk to me about it

Sketches and Drawings





neat texture too

spline like you are right.

See schematic diagram end of page 3 to get you started

(Left) General overview of scene, detail of trees; (Right) detail of tokens and obstacles

Well done

Neat. You should make a scene graph head / body.

Before that you should draw Emrys from the side, to have a 3D vision of its form.

(Above) rough drawing of Emrys (modeled after heroes demo)

Technical Outline

Not necessary; you should have an endless sloped plane. It will be easier. It is more integrated with your game. It will be visually more adapted

1. **Ski Hill**: We will model the ski hill as a cylinder or sphere so we can keep moving the sled down the hill endlessly. We will also apply a snow texture on the hill.
2. **Emrys and the sled**: We will create an Object3D to hold models of both Emrys and a sled. Emrys will be modeled based off of the cat hero in the heroes demo[1] with appropriate animations, and the sled will be a rectangular box.
3. **Trees**: We will model the trees similar to the ones shown in class i.e. composed of simple geometric figures like cylinders and cones. We will first model trees on a horizontal surface, then change the code to model trees on a hill. They will be randomly generated on the sides of the ski hill.

↳ sloped plane

4. **Tokens**: We will model them as textured thin cylinders, with a different texture for each type of token. They will appear standing up, i.e. not laid down on the ski hill. They will rotate clockwise along their vertical axis. ← part of 12

Are you planning to control (i.e. avoid) overlap and set density?

5. **Branches**: We will model them using an Object3D composed of simple geometric figures such as cylinders and cones. Add rocks and papers

Many avoids as you have many tokens to collect

6. **Buildings**: We will model a building from campus, namely Huntington Gymnasium, to place in the background. We will incorporate bump mapping to make it have rocky textures so that it appears more realistic.
7. **Skybox**: We will use blue sky with clouds as our skybox for the scene ← Texture the buildings in your skybox
8. **Particles**: We will incorporate particle effects in the scene to have a snowing[2] effect. We will also have particle effects for collisions[3] with tokens and branches.

Win / Lose / Timer: end / lose visual condit°

9. **Shadows**: We will use Three.js shadow map features[4] to handle shadows for all objects on the hill, including Emrys and the sled, trees, tokens, and branches. Shadows will be generated from a light source that moves with the sled around the hill.

lighting / fog: with controls

Not really an objective but an integrate that goes w/ 13

10. **Moving Emrys and the sled**: We will allow the user to move Emrys and the sled to the left and right with the ArrowLeft and ArrowRight keys, respectively. We will also allow the user to speed up and slow down the sled with the ArrowUp and ArrowDown keys, respectively. Finally, we will allow the user to make the sled jump with Spacebar.

make it optional: it will be great if you have time for it

up to the tree zones

Good idea

11. **Physics engine/Collisions**: We will incorporate Physijs[5] to handle the physics of the sled's movement, particularly jumps, and any collisions with obstacles.
12. **Animations**: Tokens will rotate clockwise along their vertical axis and the Emrys will move on the sled in the same direction as the user moves the sled.
13. **GUI**: A graphical user interface will be used for the HUD of the game, displaying the player's score as well as the number of lives and time remaining. It will also include buttons to toggle features such as particle effects[6] and shadows on/off[7] and change camera perspectives.
14. **Toon Shading**: Apply toon shading using Three.js features[8] on Emrys and the sled to achieve a more game-like/cartoonish look.

Great; sorry to think about it in 9.

Objectives

1. Model main objects i.e. ski hill, Emrys and the sled, tokens, branches, trees, and building according to schematic diagrams using Three.js geometry.
2. Include snow texture for the ski hill, randomly generated trees at the side of the ski hill, and blue sky skybox with clouds for a full experience of the environment.
3. Implement a shadow map to make Emrys and the sled, tokens, branches, and trees i.e. all objects on the hill cast shadows for a more immersive scene.
4. ~~Incorporate bump mapping on buildings in the scene for more realism.~~
5. Implement a particle effect to show snow falling from the sky, contributing to the aesthetic of the snowy ski hill.
6. Incorporate a physics engine, potentially Physijs, to realistically handle collisions and jumps.
7. Implement different ~~particle~~ effects when ~~the ball~~ collides with tokens and branches to notify the player of the (type of) collision:
8. Implement animations for the tokens and Emrys to make the scene more dynamic. The tokens will slowly rotate in the y-axis in place, Emrys will also move left and right on the sled when the user uses the left and right arrow keys.
9. Implement a menu through a GUI that will contain buttons to enable/disable snow particle effects and shadows and to toggle between first-person and third-person perspectives.
10. Implement toon shading for Emrys and the sled to make them look more cartoonish/game-like.

May look weird; let's do it with colors

include

it is own objective (4)

controlling overlap, w/ changing density

Implement an immersive environment w/ lighting, shadows

to make it a brighter overcast winter scene ← (3)

NOTE When the sun is not out there isn't cast shadows outside

Do you have an example of jumps? How is it implemented?

How does it look like?

sled you're right could reuse particle for branches, rocks.

Nice

that will be cool

Make more spin for Token.

Where is the camera? to see these

Perfect

Have you done some research about it? Do you have a tutorial or demo that you are planning to use → Bio & Good!

Stretch Objectives

1. Incorporating tree generators to make more realistic trees.
2. Creating a skybox using images taken from the ski hill.
3. Implementing spline curves to create more realistic branches.

Take picture of top (or find snowy sky texture)

definitely take picture of the bottom: for the finish line

create a function for branch with many angles set from an allowed range

Bibliography

1. https://codepen.io/Yakudoo/pen/oXJYxy
2. http://jsfiddle.net/3drjwj2n/
3. https://aerotwist.com/tutorials/creating-particles-with-three-js/
4. https://aerotwist.com/tutorials/creating-particles-with-three-js/
5. http://chandlerprall.github.io/Physijs/
6. https://github.com/mrdoob/three.js/issues/1690
7. https://github.com/mrdoob/three.js/issues/2466
8. https://threejs.org/docs/#api/materials/MeshToonMaterial ✓

Does it work on low-fidelity mesh?

Dear Asad, Zaila, Leslie and Jingxian.

You have conceived an enticing game! Your proposal demonstrates that you have carefully thought out all the elements (from models to interactions) to make their developments manageable.

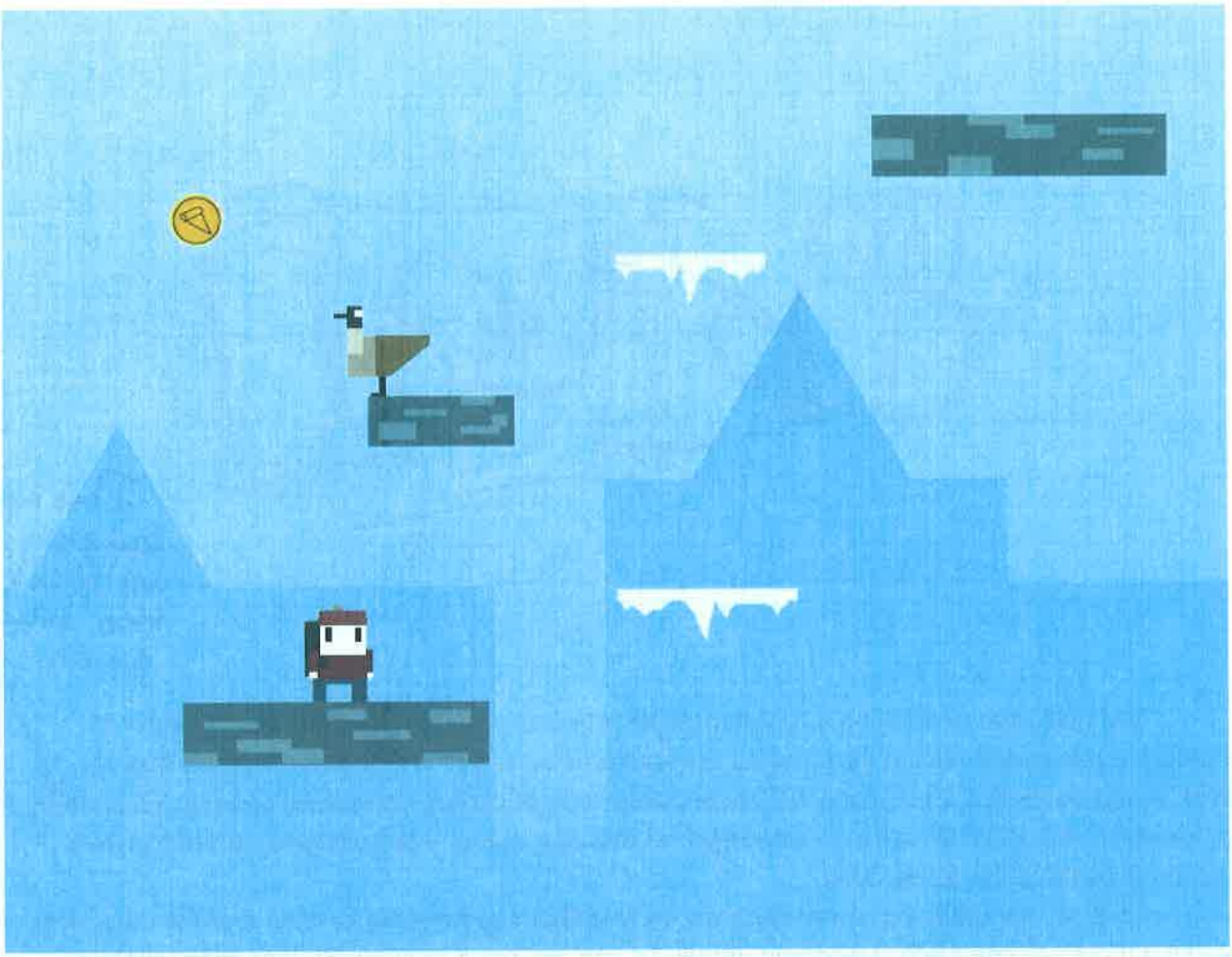
Your schematic sketches represent well your vision of the game. They are a good foundations to get you started; I an encouraging to continue to draw as you are problem-solving each part: it is graphics afterall. Proceed in stages, starting simple (constraining the feature before making it more powerful, general and complex – adding parameters, functions as the appearance and game play improve).

Your objective list is detailed, sensible and sufficiently ambitious; it only needs few revisions in term of grouping the features better. The technical outline and bibliography show you have research and planned the execution of implementing your application.

Continue
Have fun and collaborative work. Elodie

COSC 435 Final Project Proposal
Alex Rosenthal, Bryan Vaihinger, Tino Zinyama

Climbing the Hill



Nice, conveying well your vision

Statement

Our group's project is a platforming game based on the experience of climbing the hill to campus. The game will be mechanically 2D, but rendered using basic 3D forms to portray the platforms, hero, and gameplay elements. We have planned out multiple possible zones for this game to take place in, using different areas of campus as a backdrop. How many we manage to complete depends on how much time we end up having, but we plan on having at least one campus environment.

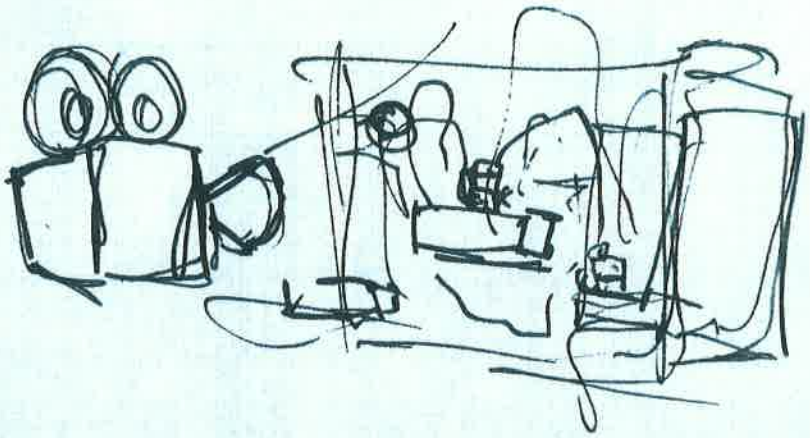
In addition to the scene, we plan on implementing multiple gameplay features to make the game fun to play. The fundamental mechanic of a platformer is movement, so satisfying lateral movement and jumping will be a priority of ours. This means responsive controls, variable jump height, horizontal lead-in and dampening, and gravity. Other interactive elements of each level will add hazards and objectives to the game. Some ideas we've come up with are:

- Goose enemies to avoid

cute (tone)

How will it be indicated visually?
Let's think about player...

- Stacks of books to climb or bounce on
- Falling ice platforms or icicles to avoid
- Strong winds that push the player
- Slippery ice that the player slides around on



This approach, layering, makes me think about the Limbo game

We also plan on making a rudimentary level editing system to make the creation of multiple levels easier. Using ascii text to lay out items on a grid is a popular method that we think would be useful. Each zone will have a pre-made backdrop consisting of basic 3D shapes arranged to represent an area on campus, and then the actual interactive section of the level can be brought to the foreground.

Overall, our ideas for our project are fairly ambitious. Keeping in mind that we only have about a month to complete everything, it may be that we have to settle for less than what we expected. With that in mind, we will build things incrementally such that we can stop at any time and have a resolved game that doesn't feel unfinished or broken.

excellent

Good thinking

Technical Outline

1) In order for controls to feel responsive, the player model's position must not only process keyboard input quickly, but update and redraw the player model on the screen as many times possible in a second.
2) We will use a physics library, such as physijs.js, to help implement gravity, and interactions with gravity such as jumping and falling. We will need to animate the player model to perform some basic jumping motion, in order to provide the player with better visual feedback of their keypresses.
3) Collisions occur in the following general scenarios. The player object interacts with a horizontal platform by landing or standing on it. The player object is unable to move

60FPS would be sufficient. The key is about the modeling/designing of the motion/animation to make the response sleek

through vertical walls. The player collides with an enemy such as a goose, or stage hazard like ice or icicles. The player object collides with some sort of bonus objective, such as a slices token. To appear polished the collisions must occur when the game objects on screen are in visually appropriate locations, e.g. touching.

4) A level editor allows us to much more quickly make levels, and the various zones within that the player can move between. The tutorial found on http://eloquentjavascript.net/ implements a basic editor that reads from a text file in order to add elements to a scene.
5) Use three.js built in library to handle the menu and options. The UI should provide the player with some sort of basic map to help keep track of their relative position within the zone, and display information such as score, time, and bonus objectives collected to the player.
6) Colgate's campus is not static, there are usually students, cars, birds, squirrels, and trees blowing in the wind. To some extent we want to capture that by providing some basic animation to background objects and scenery. We might take an approach similar to the aviator example and randomly generate clouds to move across the screen in the background.
7) We will use the three.js library's particle system to create the mentioned particle effects to give the game an increased sense of polish.
8) We will use a combination of a zone wide directional light to simulate the sun/background light, and smaller lights positioned within the scene for the rest.
9) Player model will be modeled in blender or a similar program. It should be simple and made up of basic 3d shapes so that it will fit in with the backgrounds and be easy to animate. Below is a sketch showing a possible style for the player model.
10) Geese are notable, because unlike many stage hazards, they are mobile and can react to the players movement. A very simple state based AI will likely be implemented so that the geese can do simple things such as beginning and ending movement patterns. They will also be modeled and animated similarly to the player object.

Yes: good start for rudimentary/ slightly hard-code editor

Can you explain more? The plateform moves right to a goal? or is it about the distinct levels?

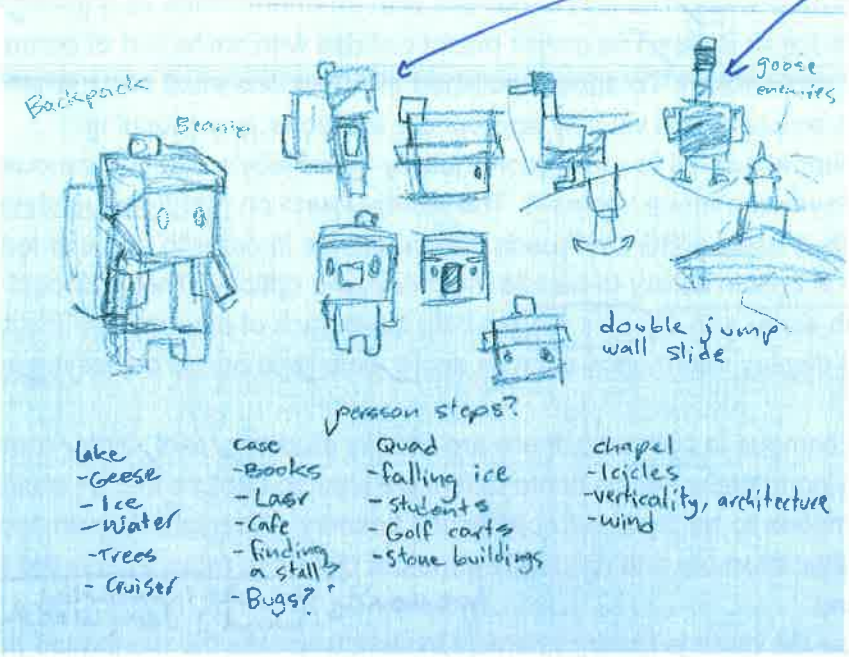
interm of ASCII that's nice for non-coder input

Are moving clouds indicating wind? Very windy days could/ can have zero clouds... maybe leaves can help represent wind too

That would be superb

Do you have experience with 3D modeling software? In the past, students encounter difficulties importing json converted models. So I want to make sure you are adequatly prepared.

You might be more successful modeling and animating them in Three.js. Let's talk further about it. You might know things I am not aware of



Objectives

1. Satisfying movement
   a. Horizontal movement and air control that is tight and maneuverable
2. Jumping
   a. Gravity that accelerates the player downward, and a jump function that sets their Y velocity upward
   b. Variable jump height based on how long the player holds down the jump key
   c. Jumping should only work when the player is on the ground, so we need to track when this is true
   d. We may add a double jump functionality for more level design possibilities
3. Collisions
   a. The player can walk on surfaces
   b. The player can run into walls and stop there
   c. The player can interact with hazards or objectives
4. Level Editor
   a. Text based level editor that converts ASCII characters in a grid to rendered objects in a grid
5. Basic UI
   a. Game should have a start and pause menu
   b. Display the user's progress and score
   c. Should feature some information and options to restart, possible other settings
6. Dynamic background to make the world feel alive
   a. Moving background elements like students or birds
   b. Simple animations of things like ~~trees~~ leaves in the wind

resistance? will produce slower motion?

Very good

required for debugging

← true objective

(make the trees the background → static)

c. Clouds moving slowly through the sky) already done

7. Particle effects and leaves
   a. Snow particle effect in some of the levels ✓
   b. Dust particle effect when running or landing from a high jump
8. Lighting Neat
   a. Lighting might change through zones to simulate passing of time through the day ✓
   b. Some game objects will emit light (buildings, cars, blue lights, etc.)
9. Player object model ← Excellent
   a. Player object should be simple enough that it will not be a pain to model and animate
   b. Style of the player object should match the rest of the game
   c. Player object will need animations for running, walking, jumping, etc.
10. Enemy Geese
    a. Geese will also need a basic model and animation
    b. Might have a very simple AI, so it will move in set patterns, or locations, given certain conditions. E.g. if a player moves to location X,Y, the goose begins moving to that position.
    c. If defeatable, possible have a particle effect to simulate bursting into a cloud of feathers

You are optimistic: it might be more work than you expect

Bibliography

Platformer tutorial: https://www.gadgetdaily.xyz/build-a-3d-platform-game-with-three-js-pt1/

2D Platformer with Text Level Editor: https://eloquentjavascript.net/16_game.html

Basic platformer example: https://codepen.io/dissimulate/pen/CqIxk

Physijs, a physics plugin: http://chandlerprall.github.io/Physijs/

Using Physijs in a basic game: https://gamedevelopment.tutsplus.com/tutorials/creating-a-simple-3d-physics-game-using-threejs-and-physijs--cms-29453

Snow particle effect example: http://jsfiddle.net/3drjwj2n/

General three.js examples: https://threejs.org/examples/

Blender tutorials: https://www.blender.org/support/tutorials/

Using tween.js with three.js: http://learningthreejs.com/blog/2011/08/17/tweenjs-for-smooth-animation/

Keep it extra

I would stay away from that approach as you seem to have an harmonious style right now; Keeping the models simple with a similar level of details will save you time and help you focus

tween.js: https://github.com/tweenjs/tween.js/

Colgate flickr account for inspiration: https://www.flickr.com/photos/colgateuniversity/ ✓

Do you have specific photo you will show during your presentation next week?

Dear Alex, Bryan and Tino,

Your proposal is strong: the image teaser gives a concrete representation of your plateform visual design: the theme 'climbing the hill' gives it a fun Colgate narrative.

You explain well the technical issues you need to solve incrementally, your research is appropriate. You made the proper comment that your project risk is to be too ambitious and took the appropriate decision to leave features out, while focusing on creating a 'resolved' game. I encourage you to pursue with this simplificato approach. One question is 'what would Blender modeling add to your game design and concepts?'

Your objective list with sub-bullet points is clear, structured, and complete (so much that you should cut some more).

Gladie