# CPSC 386: Introduction to Game Design and Production - Fall 2018

Final Project, **Crossy Road,** due Friday, 21 Dec 2018 (by 2355)

In this assignment, you will recreate the classic Crossy Road game, using the Unreal 4 Engine, assets created using a Voxel editor such as MagicaVoxel, and code written either in C++ (written to interface with Blueprints and the Unreal framework), or with Blueprints, or both.

 The 3d assets you will need will all have to be created using an Voxel editor such as MagicaVoxel, or from other 3d asset sources from Unreal or Unity. The audio resources you will need can be captured using an audio editor such as Audacity from Crossy Road running on a laptop or desktop computer, or running as an iOS or Google app on a mobile phone.
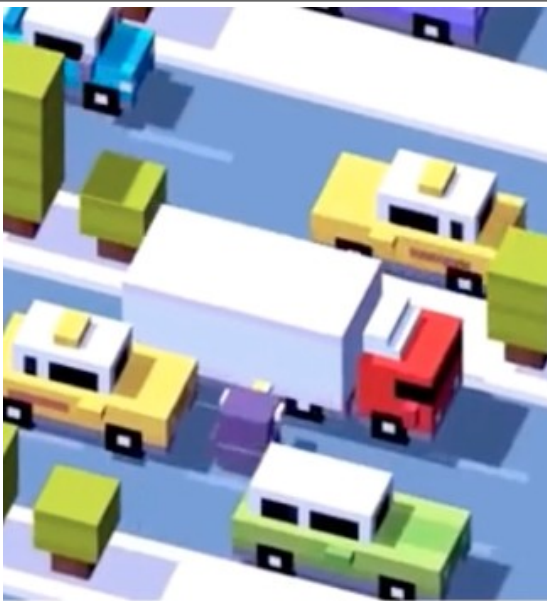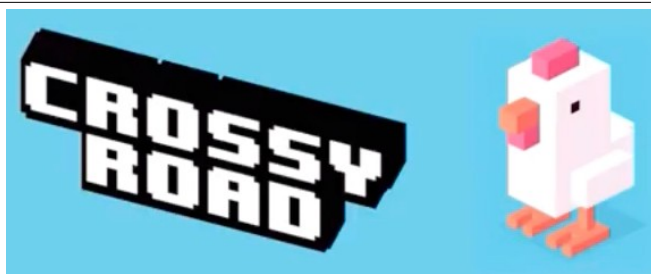
Trying to jump between the cars and the truck...
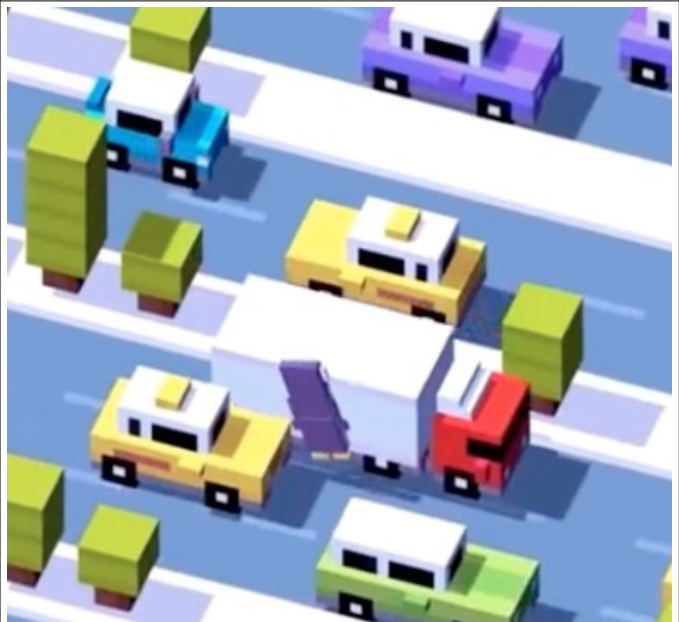
… failed… squashed on the side of the truck.

OBJECT OF THE GAME:

Live as long as possible. There is only one level, but it is dynamic and keeps being created in front of you, and destroyed behind you.

Level is made up of alternating safe areas and obstacles with either vehicles that can strike you, or water you can fall into. You can pause and wait for a safe interval to cross the obstacles -- but not too long, or an eagle will swoop down and pick you off. Unlike the Lord of the Rings or the Hobbit, Eagles are not good guys in this game.

Safe areas are grass w/ trees, bushes, and rocks). Obstacles are roads w/ vehicles, RR's w/ trains, or rivers w/ water (unsafe) and moving logs (temporarily safe)).

You can move forwards/right/left always, and backwards only within a single safe area/obstacle. (However, most safe areas are only one jump wide.)

The game designer must create the obstacles with moving cars/trucks, trains, or logs so that it is always possible for the player to get through if they are clever enough. This means that cars/trucks/trains must be timed, and trees/bushes/stumps/rocks must be placed properly.

The actor can be killed by either standing still or failing to move forward for too long (an eagle swoops down), or by being run over, or by running into the side of a car/truck/train, or by falling into a river.

The special effects that occur are as follows:

- running into the side of a car or non-oil truck: flattened version of voxel (back if running forward, front if backing into it—unlikely but possible)
- being run over by car or truck: flattened front of voxel on the highway
- being run over by/running into the side of an oil truck: explosion in three/four colors: bright, white, large, explosion (polygon), then a particle system with the actor's color, then yellow, then red (see images)
- being run over by/running into the side of a train: explosion of particles of the actor's colors
- falling into water: exploding water particles

The wizard actor also has the ability to disintegrate trees in his way, using his staff, causing the tree to char, then explode.

To create your game, you must...

Create at least four assets using a Voxel editor such as MagicaVoxel. Can work cooperatively for the rest -- coordinate scale issues (tracks for train...)

The game must contain Actors, Grass, Highways, Railroads, Water, Trees and Rocks, Cars, Trucks, Trains, Logs, Coins and the Crossy Road logo. Details below.

SUMMARY of 3d assets (voxels) your game must contain. Use MagicaVoxel to create them.
https://ephtracy.github.io

Actors, Highway Vehicles (Cars, Trucks, Roads), Tracked Vehicles (Train, Tracks, Crossing arms), Water (Rivers, Lily pads, Logs), Landscaping (Grass, Trees, Bushes, Rocks), and Miscellaneous (Coins, Logo)
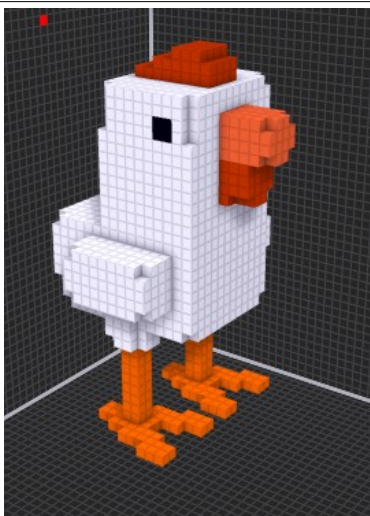
At least four 3d assets must be created by each individual. The rest can be created collaboratively. The following Google drive site has a collection of 3d assets you may use and/or contribute to:
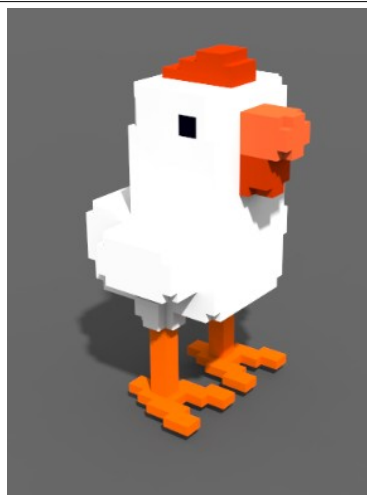https://drive.google.com/drive/folders/1HzIMNnj8iHR8-C9zfwyy5ZonDzQVjJRY?usp=sharing

ACTORS...

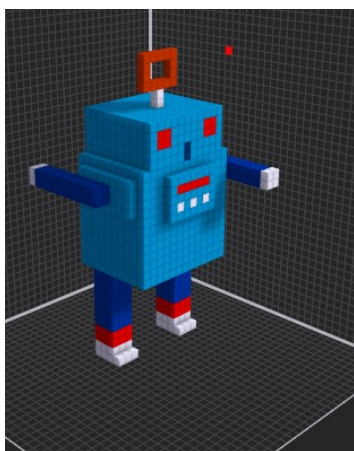- Chicken, Robot, Sheep, Fish, Frog, Steer, Wizard

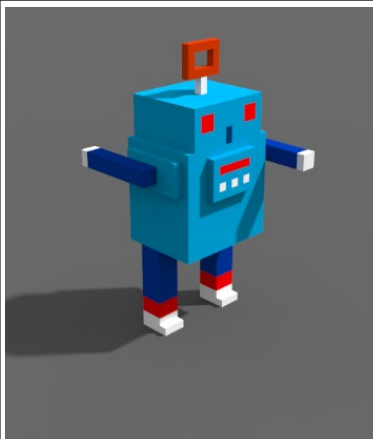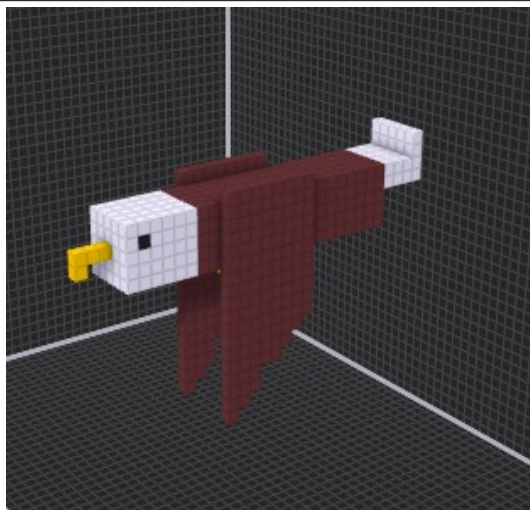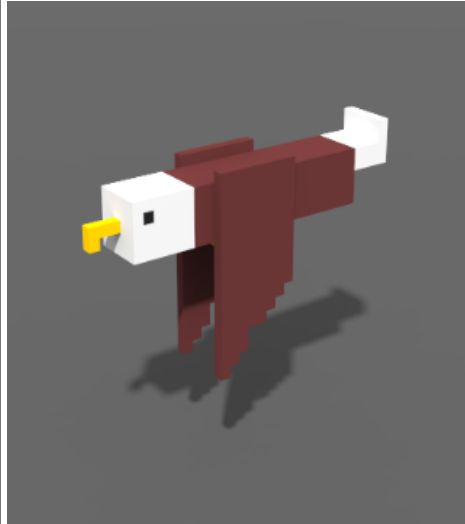|	
The Chicken, in MagicaVoxel.	The Chicken, rendered.
The Robot, in MagicaVoxel.	The Robot, rendered.

The Eagle, flapping its wings (slightly modified), in MagicaVoxel.

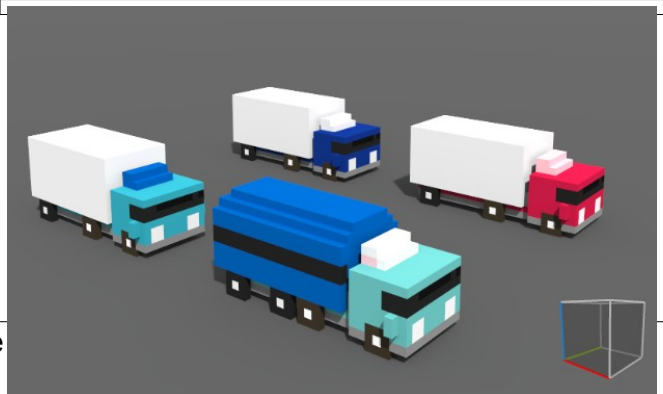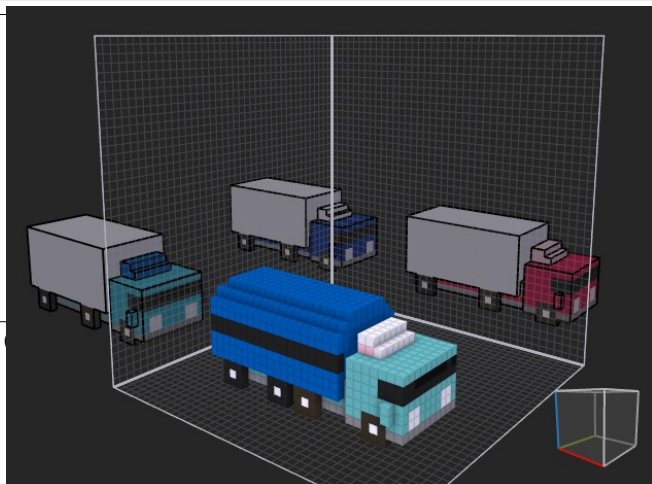The Eagle, rendered.

HIGHWAY VEHICLES and HIGHWAYS...

- Cars (Taxi, Short-blue, Long-green, orange, purple)
- Trucks (light-blue or blue or red/white trailer, and white/blue oil tanker which can explode in flames)
- Roads (1-, 2-, 3-, 4-, 5-, 6-, …, -10 lane roads). Multi-lane roads have lane markers.
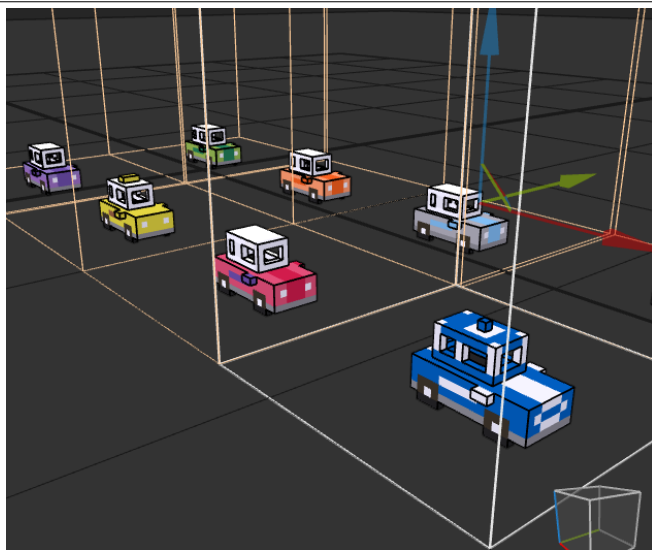
Cars, including a police car, in MagicaVoxel.
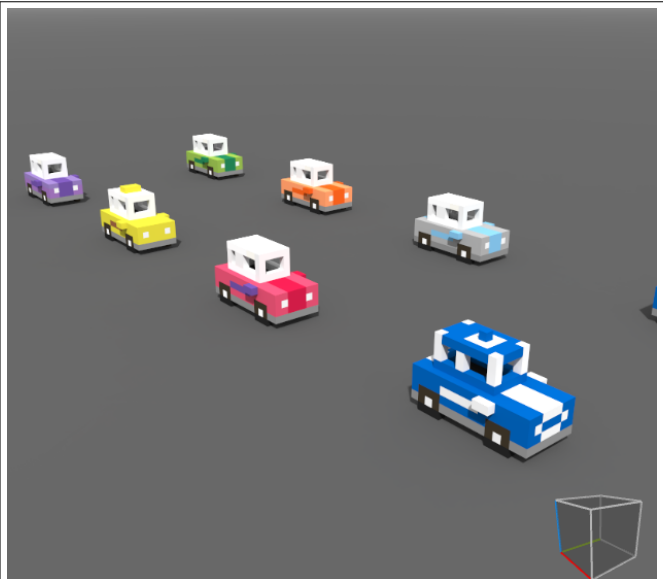
Cars, rendered.

Trucks, including an oil truck that explodes, in MagicaVoxel.	Trucks, rendered.
Four-lane highway, in MagicaVoxel.	Four-lane highway, rendered.
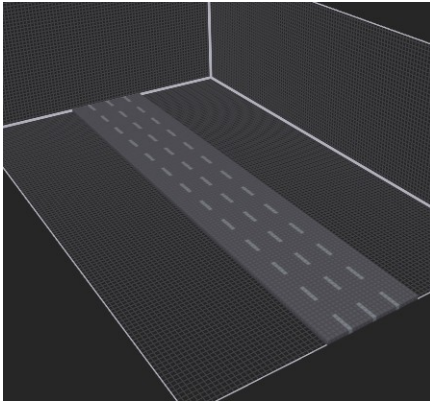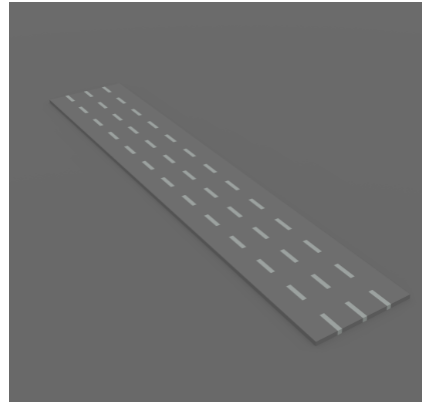
RAILROADS and TRAINS

- Tracks (1- and 2-tracks. Tracks can be between any two obstacles or safe areas.
- Train of 5-6 cars made up of a:
- Lead car, middle cars, end car

Train (lead car, middle car, trailing car) in MagicaVoxel	Train, rendered.
Train tracks and the train crossing warning in MagicaVoxel	Tracks rendered.

-

WATER

- Water w/ white reflection showing limit of range of motion, 1-, 2-, 3-, 4-, and 5- logs wide
- Logs (full (truck)-size and 2/3 size)
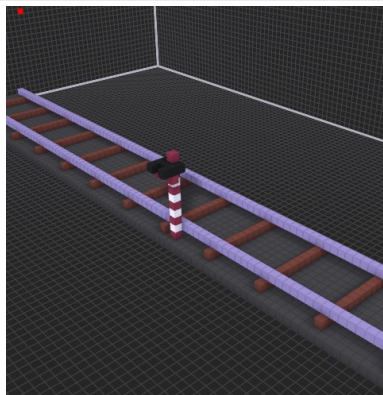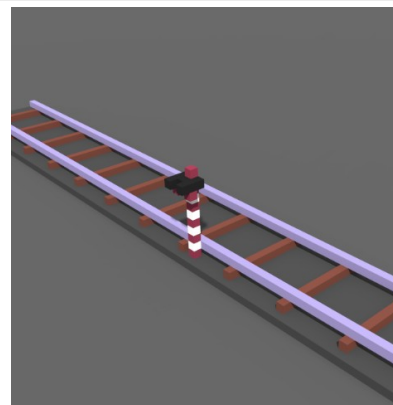- Lily pad
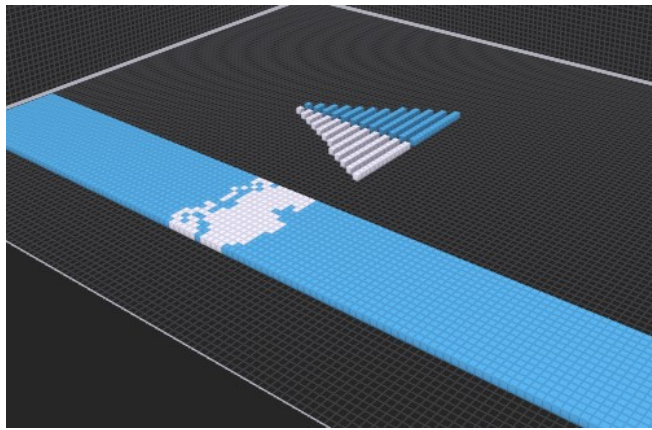
Water, with different lengths of white and blue water, to form the waterfall, in MagicaVoxel.	Water with waterfall, rendered.
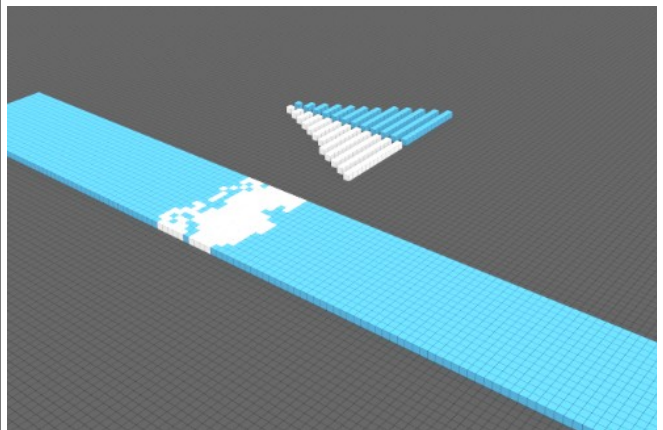
LANDSCAPING

- Grass (1- and 2-trees wide)
- Trees (tall, medium, short, and stump. Can be exploded by robot)
- Rocks (short of various shapes)

MISCELLANEOUS

- Crossy Road logo (black/white—projected) – must be rendered in Unreal4 for proper effect.
- Other effects, FedEx logo, UPS logo, etc. on trucks, also must be rendered in Unreal4.
- Coins with red 'C'
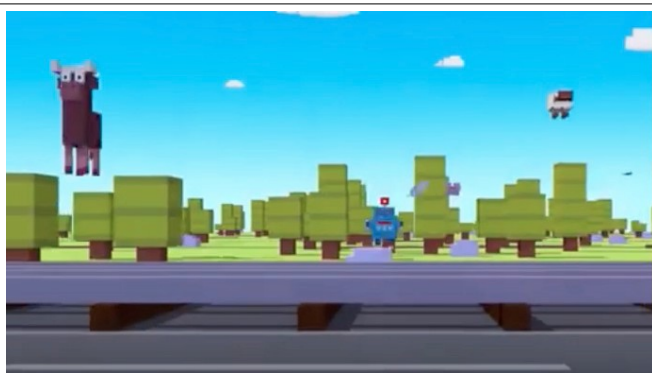
Steer, robot, and sheep (and rabbit in the background) jumping through our world of grass, highways, railroads, and rivers.	Closer view of robot and sheep.

Single and double-lane highways, with a single-track railway in the middle. Cars, taxis, and trucks going up and down the highway.
Multiple obstacles can be side-by-side, although usually grassy safe areas will alternate with obstacles (highways, RRs and water).
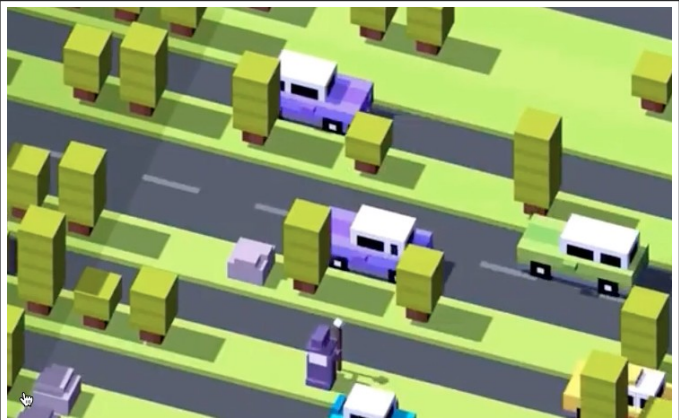
Single lane strips of grass (with rocks, bushes, stumps, and trees). Single and double-lane highways are shown here, with cars and trucks.
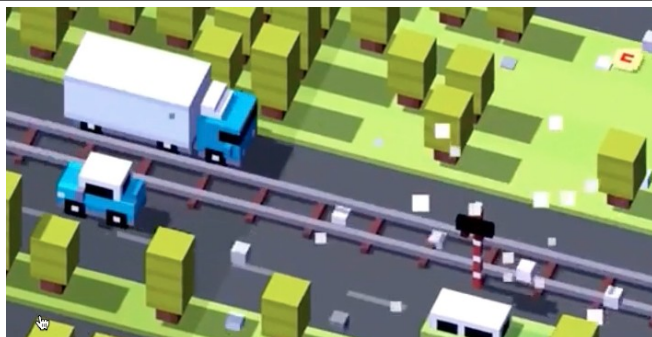
A RR can run in between two highways.

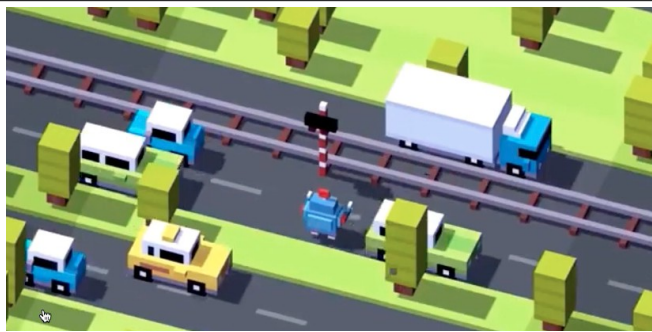Six-to-seven car trains can whiz by. The chicken has just been struck by the train, exploding into a particle system of white cubes. Several feathers also whirl in a stream of air. The chicken bawks loudly when it is struck.

The Wizard can incinerate a tree with electricity
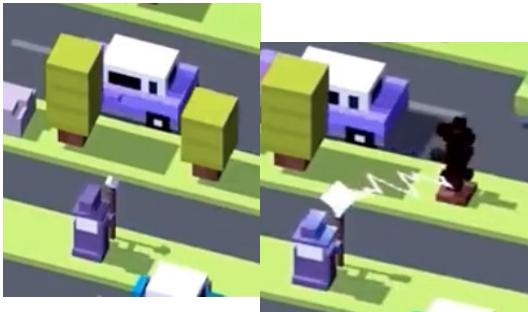
A sizzling noise is heard when the wizard chars the tree, then the tree explodes in a particle system of flames.
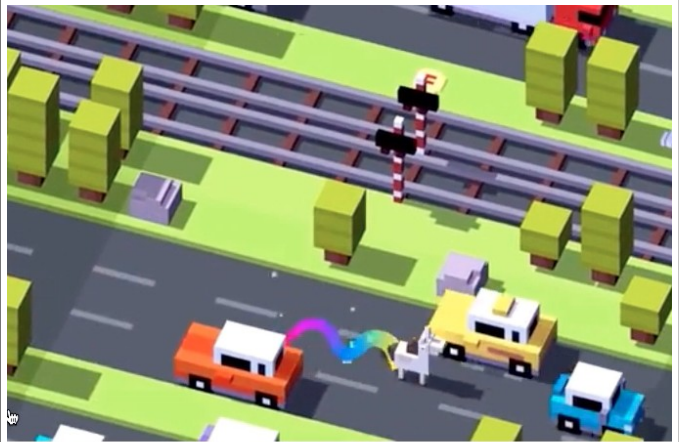
Rainbow-tailed Unicorn runs down a three-lane highway, with a two-track railroad nearby. The unicorn neighs as it moves.
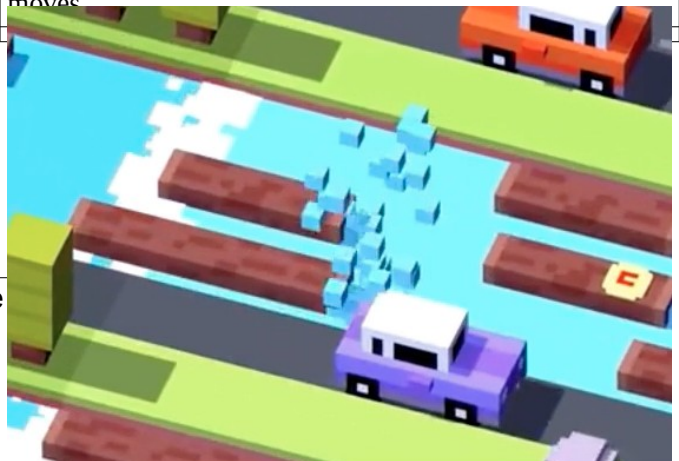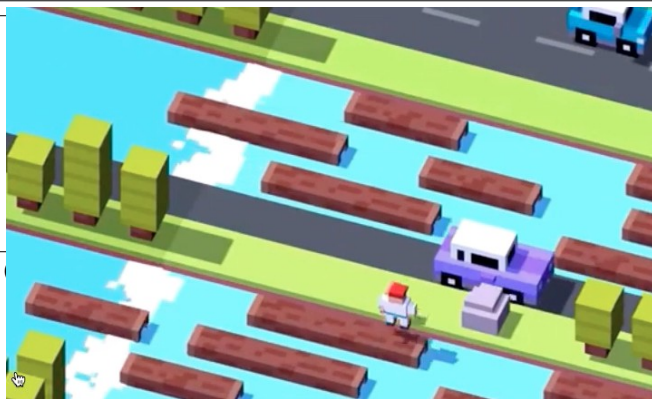
Note a 4-log stream is next to a 3-log stream and a highway, with only a 1-tree wide grass strip to pause within. The actor tries to cross the stream, and … (Note: the logs and lily pads sink slightly in the water as they are jumped upon, and make a noise (wooden chime for the log, stone-dropped-in-a-pond for the lily pad).	… a splash of particles of water explode if they fail.
  A failed crossing ends up on the side of a truck	 Before and after

1. Downloaded the game onto their iPhone or Android phone, and played it often enough to understand: game UI including keyboard feel and HUD, game aesthetics, game feel of play, game reward system (top scoreboard, current score, coins), and game failure system (being run over, exploding, water splashing, eagle snatching you up).

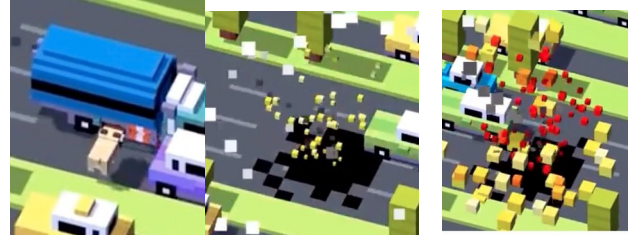2. Be able to duplicate the game to the extent that others enjoy playing their game (actors sliding on levels with one car duplicated over and over would not qualify as enjoyable). Note, other players have highly-honed standards.

3. The game has a startup screen that shows the name of the game's logo, with the logo sliding in from off screen to the center of the screen. The logo is shown at a down angle of 30 degrees (matching the angle of the safe areas/obstacles).

4. High score is shown at the top-left corner of the screen, just under the current score. Current score is the number of jumps taken so far. Coins collected is shown at the top right-hand corner. If a new high score is reached, it should be displayed as new high score in a popup window. Also, if close to a new high score, it should be displayed on the grass, water, highway, or railroad, with the text angle matching the safe area or obstacle.
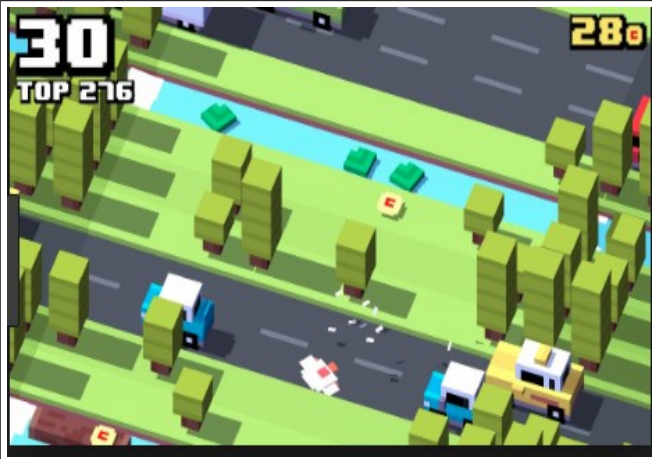
Note the dead chicken on the highway. It has been run over.

5. Implemented all of the 3d actor assets in the game (Chicken, Bunny, Fish, Frog, Pig, Robot, Sheep, Steer, Unicorn, and Wizard). Minimum number of assets for an individual is four, if cooperating with others for the assets. This includes implementing the safe areas, obstacles, and rewards.
6. Implemented all of the safe area, obstacle and reward 3d assets in the game (grass, highways, railroads, water, coins), complete with their contents (trees and rocks), (cars and trucks), train w/ multiple cars, logs and lily pads). The contents should be of various types for interest and varying difficulty. Minimum number of assets for an individual is four, if cooperating with others for the assets. This includes implementing the actors.
7. Implemented the ability of the game to dynamically generate the contents for each obstacle/area.
8. Implemented the actor's movement through the game (crouch, jump, collide with immovable objects –rocks, trees, train crossing, and collide with fatal objects—cars, trucks, trains, water, and eagles. Note, collision must use SWEEP, **not** teleport in unreal 4. This will require keyboard handling that holding down a movement key will cause the actor to crouch until the key is released.
9. Implemented the game's ability to dynamically generate/destroy obstacles and safe areas in front of/behind the actors as they move.
10. Have created particle systems with appropriate colors and sizes for running into/being run over by trains, exploding oil trucks (and actors), or splashes of water, and connected them to the game.
11. Gave different noises and frequencies of occurrence to the various actors (cluck for chicken, meow for cat, quack for duck, moo for cow, boing for robot, baa for sheep, ribbit for frog, floppy noise for fish, neigh for unicorn, oink for pig)
12. Used a sound editor such as Audacity to record the sounds of the game as you play. If you play the app on your phone, with the volume at maximum, in a quiet place (first thing in the morning in your home for example), you will be able to capture all of the sounds, as follows:

Moving sounds (moving the chicken)

Chicken (other actor) sounds (chicken clucking every so often, can be different frequencies) Chicken dying sounds (loud b-gawk! Sound when killed by running into car/truck/train, being run over…, picked up by eagle, falling into water, …)

Cars/trucks sounds: tires on highway plus truck engines (soft, trucks are louder), long/short

beeping occasionally (beep/honk)—can be close or far away, beeps can be different frequencies, long doppler-shifted horn of sports car occasionally, police-car siren, explosion of oil tanker if hit by chicken

Game over/beginning sounds (ascending bloop, bee-boop descending)

Train sounds (bell of train coming, train whooshing by)

Water sounds (stepping onto lily pad/log (stone dropping in water, creaking stair), falling into water)

Picking up coin sounds (ascending series of xylophone tones)

- truck coming
- chicken run over
- bell of coming train
- game over
- train going by
- start of game
- car beeping

13. Connected the sound files to the activities in the game: actor moving, different kinds of actors making sounds (e.g., chicken, duck, sheep, pig, ...), cars/trucks moving and beeping, train coming/passing, stepping onto logs/lily pads, being run over, eagle shrieking, game over/game beginning, picking up coins.
14. Ensure that every python source file shows a green checkmark in Pycharm (passes all PEP 8 requirements).
15. Push the contents of your project to a new GitHub repository using a git client (e.g., the git command-line client, GitHub Desktop, or GitHub for Atom). Do not submit files using drag-and-drop onto the repository web page, and do not push this assignment to the same repository as your previous homework assignments.

Submission

Turn in the code for this project by uploading all of the Python source files you created, the images directory, and the sounds directory to a single public repository on GitHub. While you may discuss this homework assignment with other students. Work you submit must have been completed on your own. To complete your submission, print the following sheet, fill out the spaces below, and submit it to the professor in class by the deadline. Failure to follow the instructions exactly will incur a 10% penalty on the grade for this assignment.

CPSC 386 Final Project, due Friday, 21 Oct 2018 (at 2355)

Your name _____

Repository `https://github.com/`_____/_____

Verify each of the following items and place a checkmark in the correct column. Each item incorrectly marked will incur a 5% penalty on the grade for this assignment.

Completed	Not Completed	**Crossy Road**
✔	☐	Have Crossy Road installed as an app on their mobile phone.
☐	✔	Game has startup screen with Crossy Road logo sliding in from the upper right at a down angle of 30 degrees.
☐	✔	Implemented the game's HUD (head's up display) showing the high score, current score (number of jumps), if this is a new high score, and coins collected.
☐	✔	Implemented at least four of the 3d actor assets in the game in MagicaVoxel (Chicken, Bunny, Duck, Fish, Frog, Pig, Robot, Sheep, Steer, Unicorn, Wizard, or custom actors). If you have custom actors, list them here: _____, _____, _____, _____. The rest can be developed collaboratively.
✔	☐	Implemented the safe grassy area 3d assets in MagicaVoxel, **including code to populate them w/trees and rocks**. Trees/rocks should allow the actor to pass.
✔	☐	Created highway obstacle (1-10 lanes) area 3d assets in MagicaVoxel—w/**code to populate them with cars/trucks, and control their movement**. Cars/trucks are timed to allow the actor to cross. Multi-lane roads must have lane markers.
✔	☐	Implemented railroad obstacle area 3d assets—1-2tracks—w/**code to populate them w/trains/crossing arms and trains, and control their movement**. When a train is coming: ring a bell, light crossing arms brightly, and flash left/right. Trains are 5-6 cars long, w/ leading, trailing, and interior cars. Trains should fit on the tracks. Multi-track trains should allow the actor to pass.
✔	☐	Implemented the water obstacle area 3d assets in MagicaVoxel—1-6 logs wide, randomly selected—with code to populate them with logs/lily pads, to control their movement, and control the waterfall's appearance. Logs/lily pads are timed to cross safely without being carried out of the water's range of safety. Actors carried into the waterfalls should cause the actor's death.
✔	☐	Imported all actor, safe area, obstacle and miscellaneous 3d assets into Unreal 4, and rotated and scaled them to their proper proportions.
☐	✔	Correctly implemented crouching and jumping with delay with Blueprints or in C++, so the actor crouches as long as the arrow key (left/right/up/down) keys

☐		are pressed, but jumps immediately when it is released.
☐	✔	Correctly implemented the actor's movement through the game, so it jumps from spot to spot, turns to jump in the correct direction, makes jumping sounds (and occasional actor sounds), refuses to move into immovable obstacles (trees, rocks, train crossing arms), and dies when colliding with fatal objects.
☐	✔	Implemented the particle system for the splash of water when the actor falls in, and the explosion of the actor when it is run over, or runs into a train or oil truck. Correct colors should be emitted, according to actor and oil truck flames.
☐	✔	Implemented the eagle asset, imported it, and written the code that controls its swoop down and carry away of the actor if there is no forward movement for a short time, or if the actor moves backwards several times.
✔	☐	Implemented the dynamic generation/destruction code for allowing the level to be continuously populated as the actor moves forward in the world. Note: the actor cannot move backwards to areas that have already been destroyed.
✔	☐	Used Audacity to record the music and game sounds, and implemented them. Actor sounds (cluck, quack, flopping, ribbit, baa, neigh, boing, _____) when moving, and when dying (louder, more shrill versions of above). Horn sounds, bell for train crossing warning, swoosh when train goes by, eagle shrieking.
☐	✔	Others (at least three) have played your game and signed off on it as fun.
✔	☐	Project directory pushed to new GitHub repository listed above
✔	☐	Project directory has been pushed using a GitHub client, not by manually dragging-and-dropping files onto the GitHub web page.

comments on your submission

Lane spawner is implemented with working lanes.
The lanes delete themselves dynamically; however, the version
that was pushed had those nodes disconnected but still present.
(May be reconnected in a later commit)
Audio works for train, train signal as well as the lights for it, and
works for signaling incoming trains. All lanes spawn their components
such as cars and trains, but move them on the first iteration
 without audio and lights only, afterwards they work as intended.
Got a lot done in the 2 days before due date, but will continue to work
on the project and commit afterwards.