August 2011

Crow Goes Hunting

Alfreda M. Smith
Worcester Polytechnic Institute

Beth Marie Hankel
Worcester Polytechnic Institute

Christopher John Daley
Worcester Polytechnic Institute

Crow Goes Hunting

A Major Qualifying Project
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
by

____
Alfreda Smith

____
Beth Hankel

____
Chris Daley

Date: August 11, 2011

Approved:

____
Britton Snyder, Advisor

Abstract

Crow Goes Hunting is a 3D Narrative of a poem by the same name, created in the Unity engine. The project takes the viewer on an illustrated journey through Crow's discovery of the destructive power of words.

Contents

Introduction

In order to create a project that suited the strengths of the team, a clear list of goals the MQP needed to achieve was created. The most important consideration was application of the individual skills that every member of the team possessed. The team was comprised of an animator, a character modeler, and an environment modeler. It was necessary to play to each member's strengths, and create a work that would be strong without a programmer. The second consideration was the desire to create a work that would inspire and touch those who interacted with it. It was also clear that the team wanted to repurpose our knowledge of game technology to create something unique. The final consideration was that the creation should be web-based, and viewable by anyone with a computer.

After taking these factors into consideration, it was decided that the team would translate a poem into a 3D medium. The ideal work would have stunning imagery that allowed a lot of creative freedom, while also providing a challenge. A team member had an extremely interesting collection of poems called *Crow*, by Ted Hughes, and his writing was surreal, morbid, and captivating. One of the poems within, "Crow Goes Hunting," was chosen for its seemingly manageable scope and room for creative license.

Interpreting the Poem

In the creation of any narrative, it is necessary to lay out all the visual pieces to make a clear, understandable story. The initial storyboard was just that – a basic, unadulterated version of the poem, drawn out scene by scene. After some refinement in Photoshop, redrawing the panels and cleaning up facial expressions, the storyboard was used to set up the environments and shape the characters. When implementation in Unity began, technological issues brought a need to modify the story as it stood, swapping out scenes created to convey the poem into ones specific for Unity and achievable with the technology available. The final storyboard read horizontally with eight frames a page, spanning five pages.

When interpreting the poem, heavy emphasis was set on the grittily prehistoric, creationism feel of the *Crow* series of works. As assets were developed, the style evolved from modern-era forms to fit a darker theme. Assets were then updated with their darker counterparts, including replacing the shotgun with a mutilated word lizard and a standard cornfield crow with a six-eyed monster.

Art

The programs used to create the art portion of *Crow Goes Hunting* were Adobe Photoshop CS5[1], Autodesk 3DS Max 2011[2], Autodesk Maya 2011[3], ZBrush 4[4], L3DT[5], and Terragen[6].

Creating the Characters

Crow

The design of the crow, as the main character of the poem, was an important stylistic choice that would ultimately drive the look of the entire project. The poem was a very metaphorical and supernatural piece, and the crow was designed with that mystical, unnatural sense in mind. However, the initial concept stage was used to present as many ideas as possible, from cartoonish, goofy-looking designs to gritty and realistic styles. The final design mixed both realism and mysticism in a dark and foreboding multi-eyed form.

[1] http://www.adobe.com/products/photoshopextended.html
[2] http://usa.autodesk.com/3ds-max/
[3] http://usa.autodesk.com/maya/
[4] http://www.pixologic.com/zbrush/features/zbrush4/overview/
[5] http://www.bundysoft.com/L3DT/
[6] http://www.planetside.co.uk/

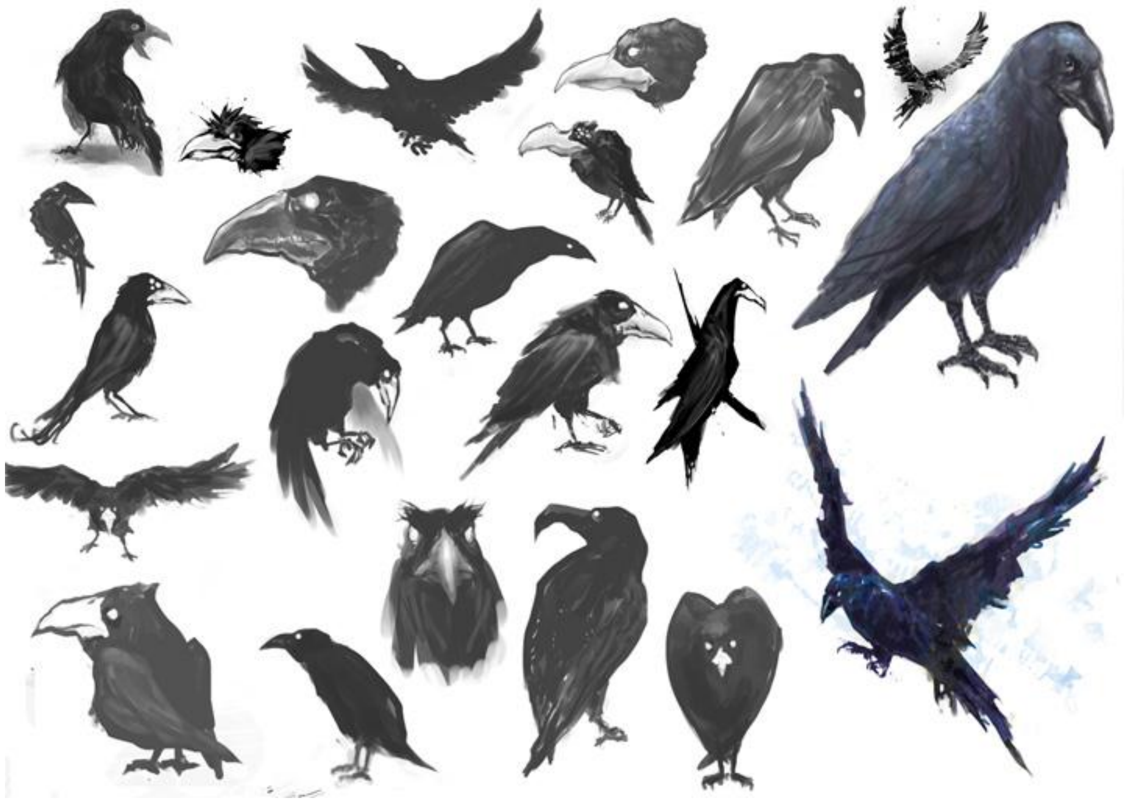


Figure 1. Initial concepts for Crow, by Ryan Chadwick.

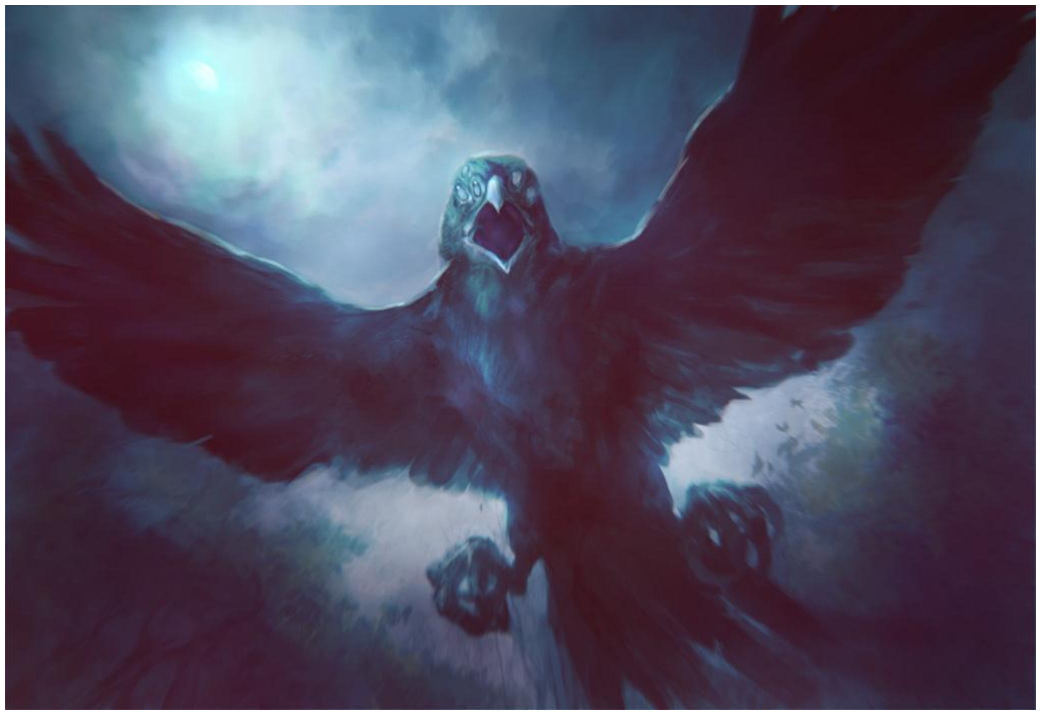
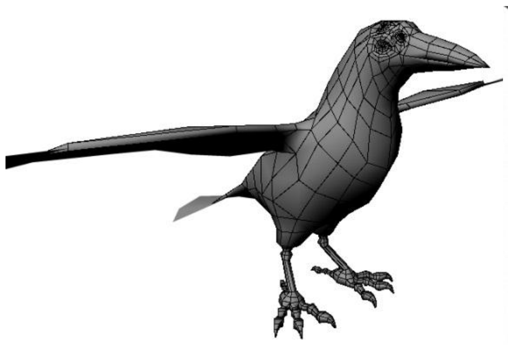
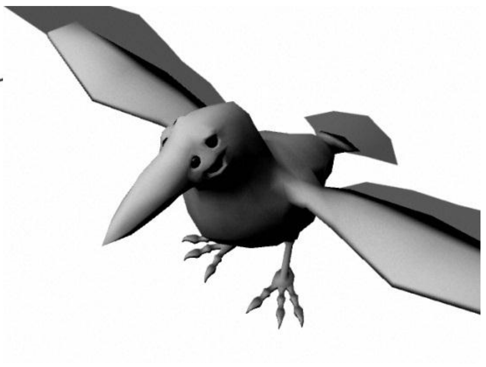
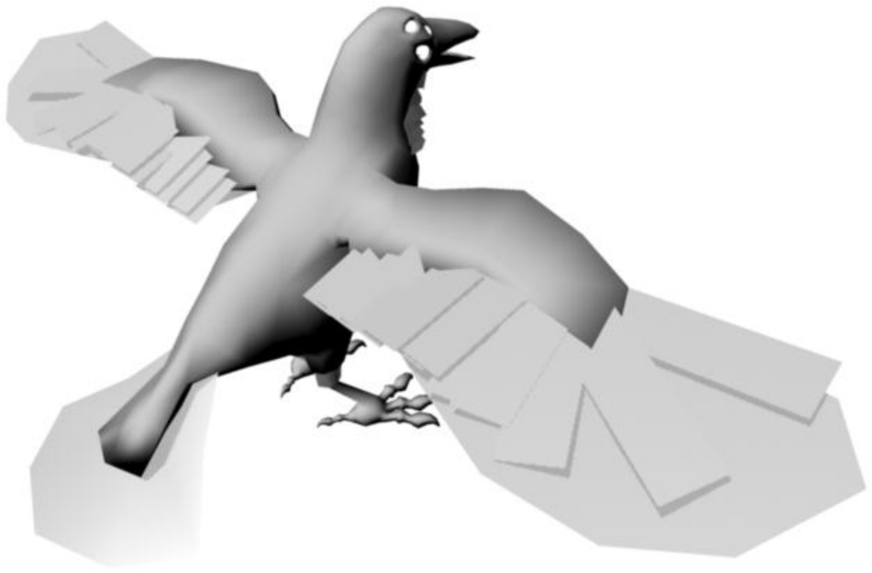


Figure 2. The final concept for Crow, by Ryan Chadwick.

The low-poly basemesh for the crow was made in Autodesk Maya in a neutral pose suitable for high-resolution sculpting as well as animation. Because the final project is rendered in real-time in the Unity engine and not pre-rendered, the crow model was created as low poly as possible without sacrificing edge fidelity in order to be as efficient as possible. For example, the individual feathers on the wings and tail of the crow were not modeled out entirely, but rather, they were created using flat planes with an alpha transparent texture to provide detail.

The crow textures were created with a mix of photo resources, hand-painting and simple image manipulation. In order to keep a realistic style, most of the base texture was taken from images of crows, a variety of other bird species and single feathers. Details and basic lighting were painted right into the texture by hand to give it depth and variation. The diffuse and full-color specular maps were both generated in this fashion.

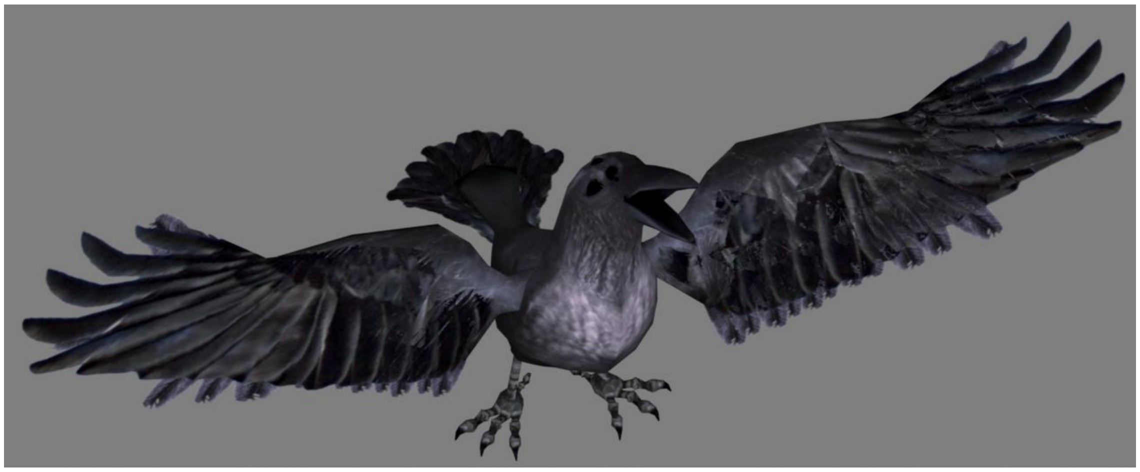


Figure 3 The final crow model.

To generate a normal map, the base mesh was imported into Zbrush. Small details, such as feathers and wrinkles, were sculpted into the high resolution mesh using the clay tubes brush. The result was a subtle normal map that provided some detail but left the major shapes and forms of the crow untouched.

Lizards

In “Crow Goes Hunting”, Crow imagines words to do his bidding. The specific words he had in mind are never mentioned, but referenced as “resounding” with “clear eyes” and “good teeth”. The word lizards were designed with strong, curved lines, fleet feet and intelligent eyes.

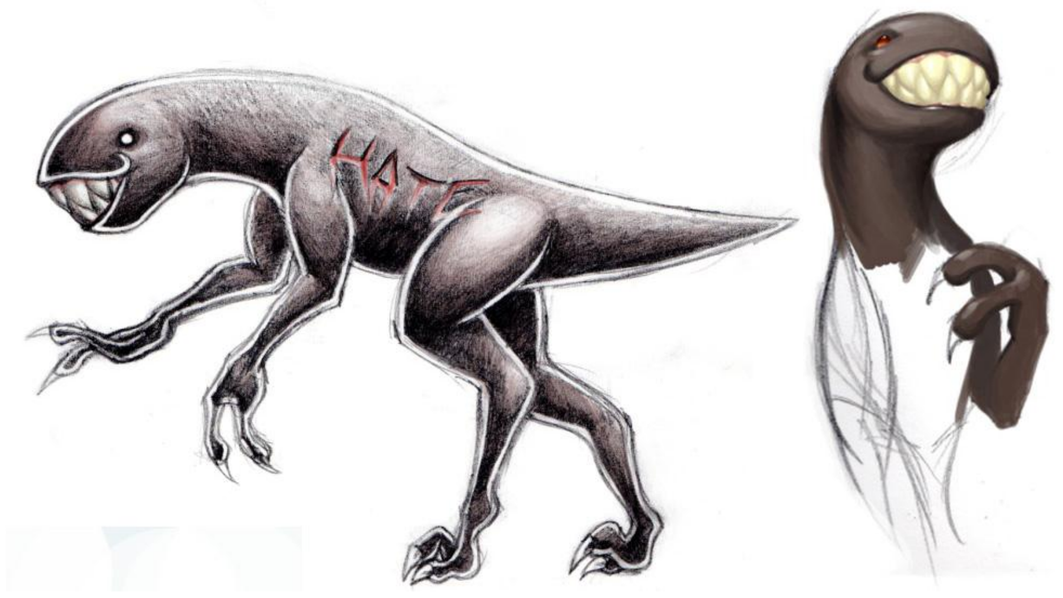


Figure 4. Concepts for the word lizards.

Rather than creating a low-poly base model, the lizard model was started with a simple ZSphere structure in Zbrush. The tool allowed the user to not be limited by poly count or polygonal shapes, and instead allowed the user to draw a shape with connected spheres of all shapes and sizes. This made Zspheres a perfect tool to easily rough out the shapes of the model to later be turned into a mesh.

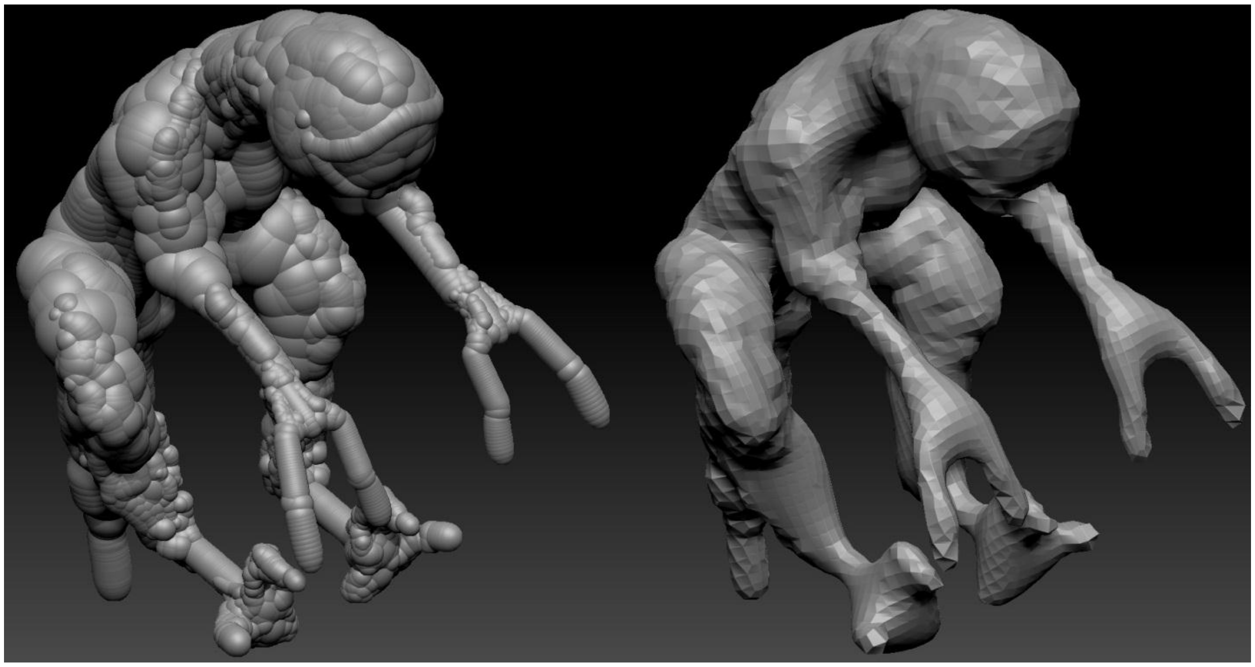


Figure 5. Left: the original Zspheres model. Note the "bubbly" structure. Right: The base mesh created from the Zsphere model.

After turning it into an editable mesh, the basic shape was refined with the move tool, and details were later added in with the clay tubes brush. A number of anatomical changes were made to the model, most notably the deletion of the "solid" teeth and the creation of separate, spaced apart teeth. The basic forms were also modified to become more reptilian .

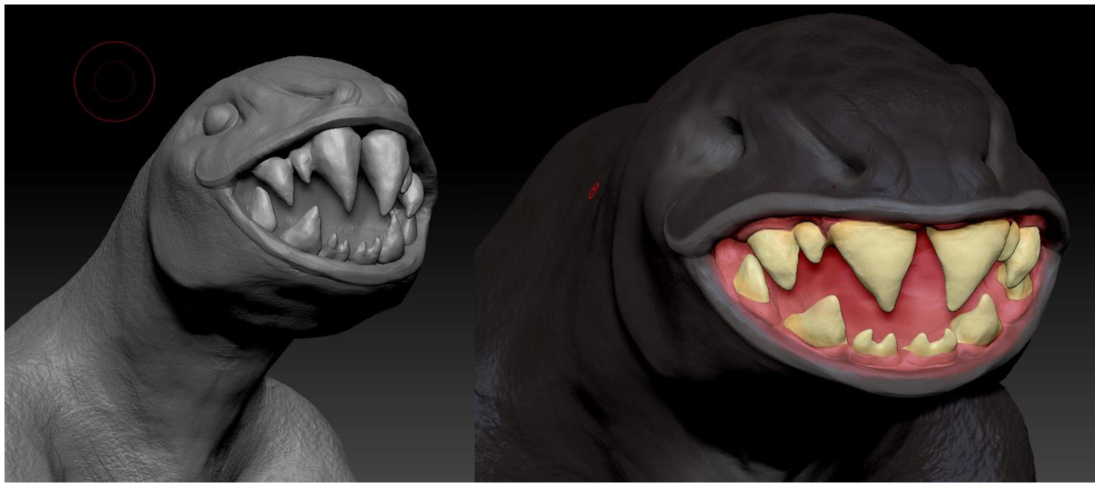


Figure 6. Left: Lizard head before reshaping. Right: Lizard head after reshaping.

The straight edges of bone structure and the curves of the muscle masses were emphasized to create believable anatomy. The entire mesh was retopologized, which removed stretched geometry and attached the teeth to the mesh. The UV maps were automatically generated in Zbrush, and the model was then painted using Polypaint for a seamless texture.

The words assigned to each lizard were chosen for their powerful meanings and negative connotations. The words the team decided to use were acrid, bleak, cheat, chaos, and dread. The words were carved into the lizard model in Zbrush, creating five unique models. The words each had a bruised effect painted in the texture to help emphasize the carvings. Superficial wrinkles were added and highlighted using Mask Cavity to help increase the realism of the lizard.

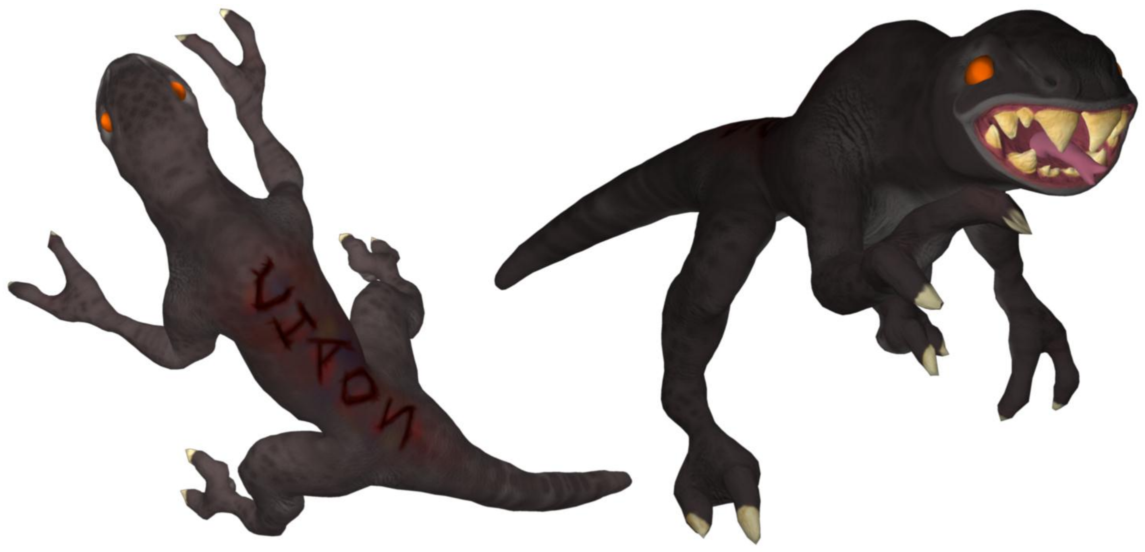


Figure 7. The final lizard model.

Hare

In the poem, the hare is referred to as a simple rabbit, chosen by Crow for the sheer joy of taunting another creature with his words. In concepts, the hare was depicted as a small, unsuspecting white rabbit, the sole innocence in a dark world.

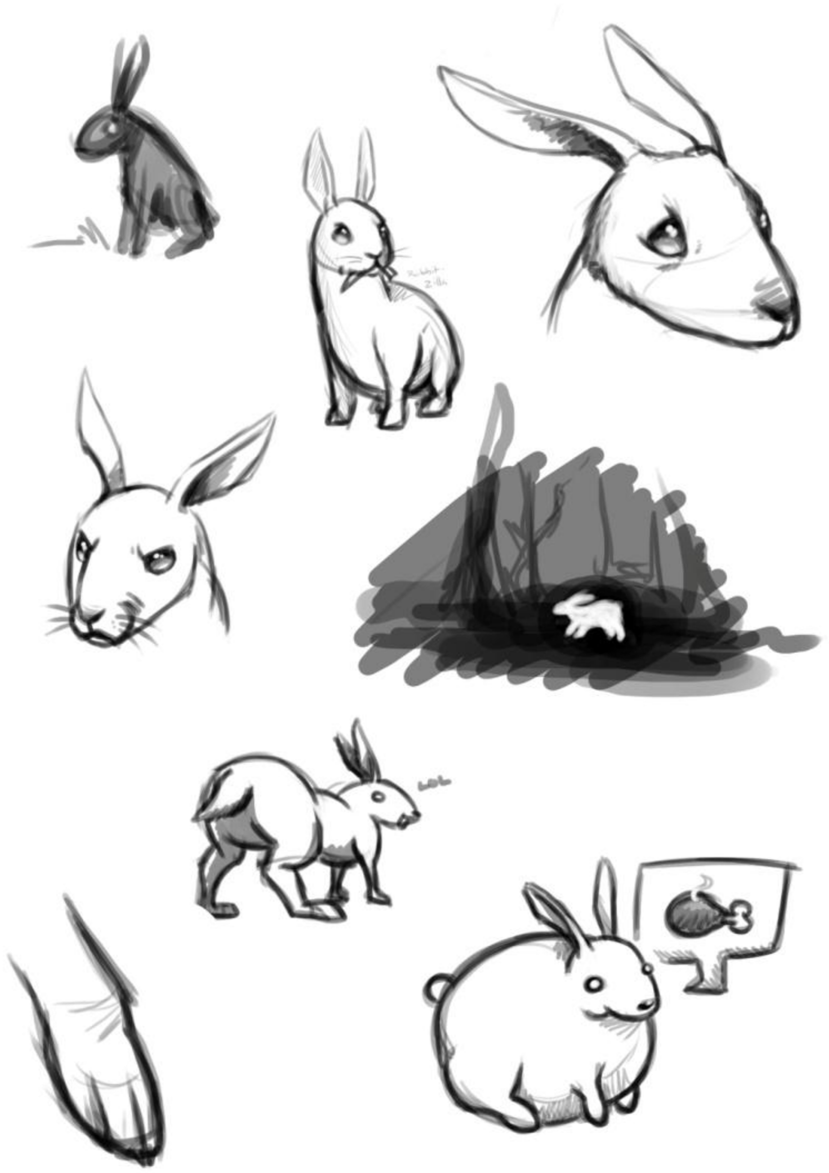


Figure 8. Concepts for the hare.

To paint the hare, custom fur brushes were created in Photoshop. Photoshop's custom brush system worked like an alpha texture. Details painted in white showed on the brush, grey was slightly opaque, and black was transparent. To create a brush for fur, a rendition of hair was painted onto a black background. In order to paint continuously, the texture had to be offset and made seamless. This brush, pictured below, constituted the majority of the hare’s fur painting, and the original, non-repeating brush was used for individual touch ups.

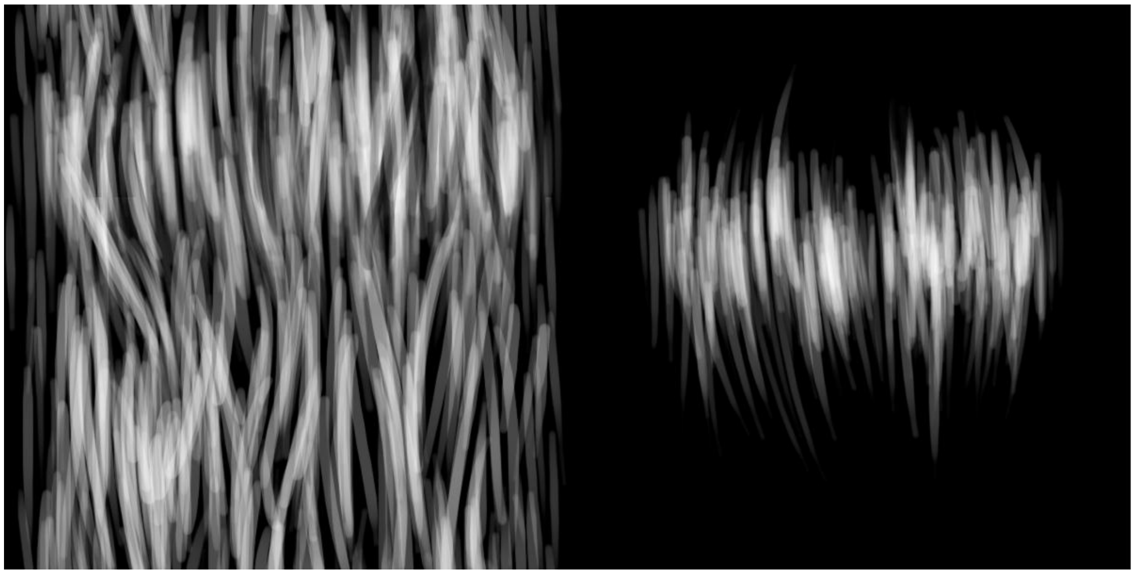


Figure 9. The custom brushes created to texture the hare's fur. Left: The seamless texture. Right: The original texture.

The model was created in much the same way as the lizard. It began as a simple ZSphere base. Due to oversimplification, when the Zspheres were converted into a mesh, the head and ears fused and geometry was lost. After the mesh was corrected, the anatomy was tweaked to aid realism. However, it was not preferable to make the bunny hyper-realistic, as the innocent aspects of the hare could not be sacrificed.

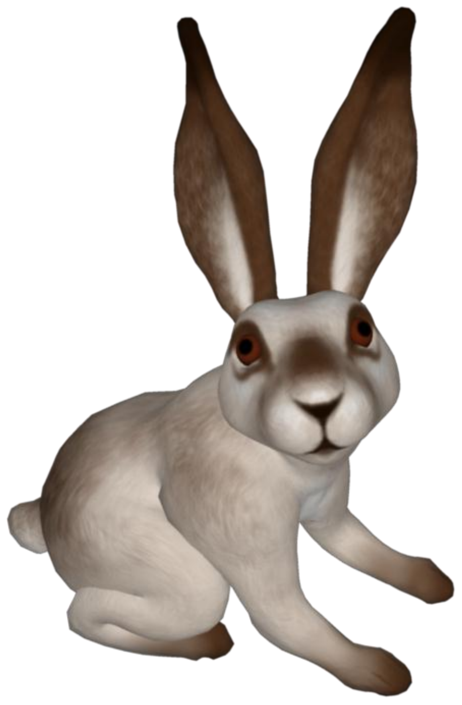


Figure 10 The final hare model.

Starling

The poem referenced a flock of starlings that was shot down by Crow. Because the starlings were going to be in a group and only visible from a distance, their model was created simply. Unlike the Crow, the starling did not require complex detail and animation, so the wings were created solid with feather detail modeled on. In order to read well at a distance, the texture of the starling was designed with bright colors that were high in contrast.

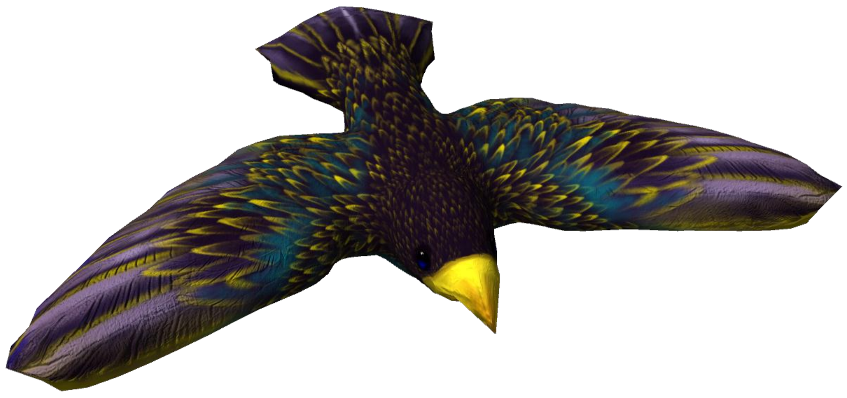


Figure 11. The low-poly starling model.

Creating the flock of starlings was achieved through an extremely creative use of Unity’s particle effects. A render of the starling was used as the billboarding texture – otherwise known as the particle. By default, Unity particles continuously spawned and decayed to give a realistic effect for situations like fire and smoke. The settings were changed to allow the particles to only spawn once and never die so that the birds would not blink out of existence. Additional speed and velocity settings were changed in order to achieve a look similar to that of a spiraling flock of birds. Using particle effects in lieu of a flocking system with 3D models was great for the project’s performance because it created a much lower poly count in the scene.

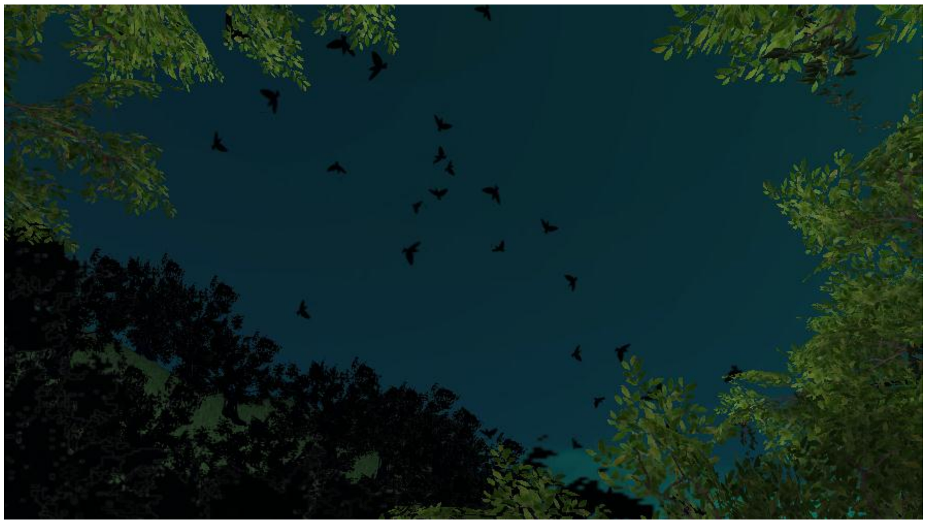


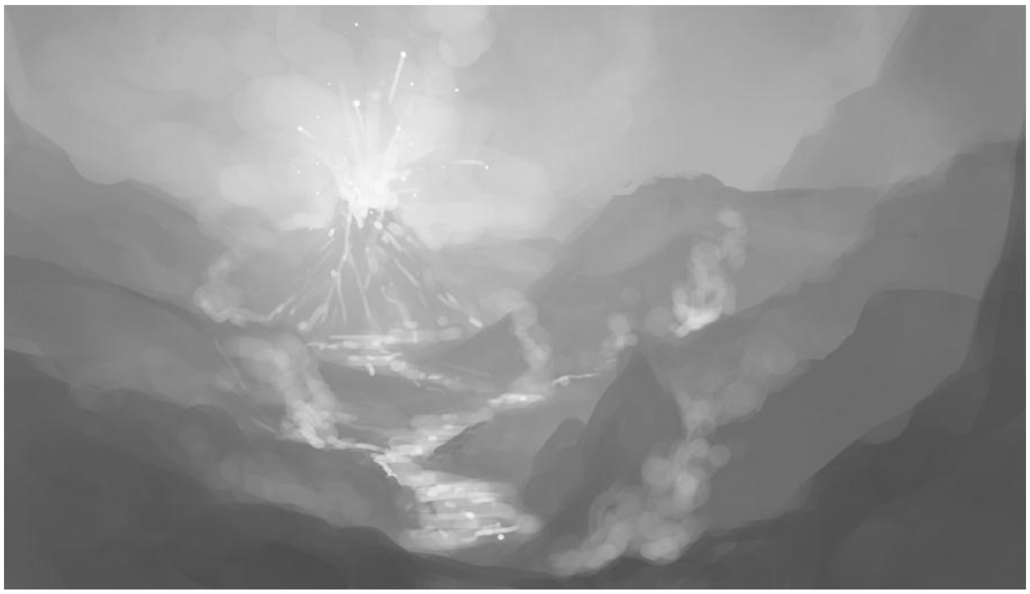
Figure 12. The flock of starlings seen through trees

Creating the Environment

Two worlds were built for the project. One was the environment that took place before an earthquake, and the second was after the earthquake had wreaked havoc on the area. It was designed to be similar to the preliminary scene, but filled with much more violent colors and many destructed elements.

Conceptualizing the World

While the design of the crow was largely influenced by the tone of the poem, the overall look of the project needed to be presented in a cohesive manner. The metaphors in the poem jumped around a lot, from military structures, to animals and natural disasters. In order to capture the wide ranging subject matter of the poem, a mystical, ancient Celtic theme was agreed upon. Certain pieces, such as the shotgun and missile, were presented in more of a magical way so as to not discourage the reader with a jarring difference in technology. By being presented as an ancient and magical story, the piece could draw upon the reader's familiarity with other old fables or parables.



Generating the Terrain

The first step to creating the *Crow Goes Hunting* world was building the world. A program called Large 3D Terrain (L3DT) was used to generate the terrain. The maps were created with a high occurrence of cliffs and terraces to add interest and variety to the landscape. L3DT automatically created a variety of texture maps, but the only one required for the project was the height map, pictured below. The initial heightmap was not ideal, so it was edited and brought to its final iteration.

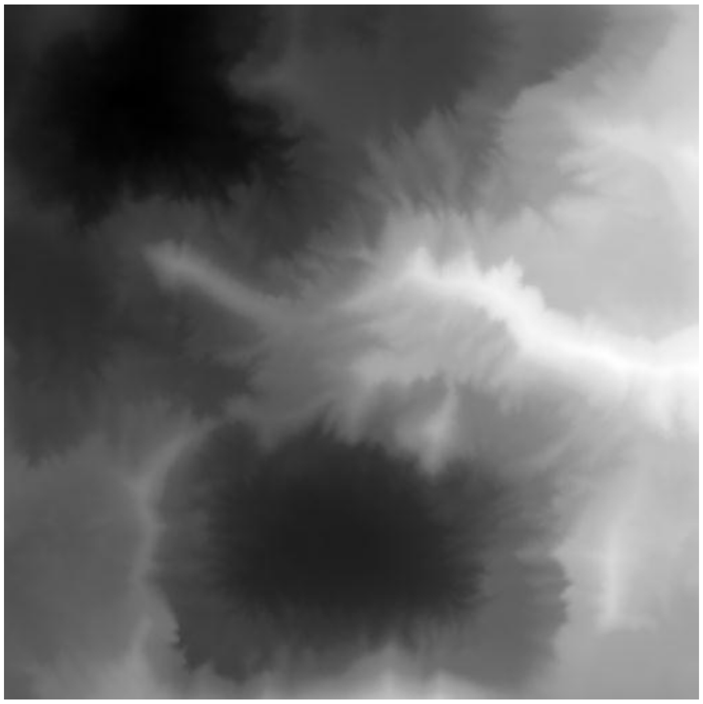
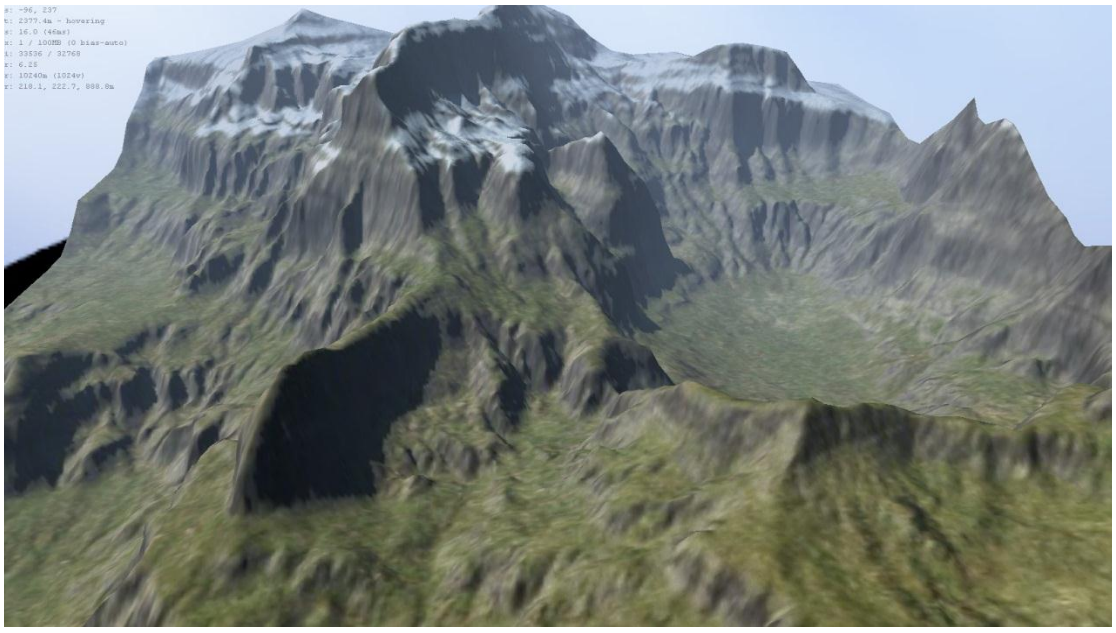


Figure 13 Heightmap



```
s: -96, 237
t: 2377.4m - hovering
s: 16.0 (46ms)
x: 1 / 100MB (0 bias-auto)
i: 33536 / 32768
r: 6.25
r: 10240m (1024v)
r: 218.1, 222.7, 888.8m
```

Figure 14 Translation of the height map in 3D

Creating the Skybox

The skyboxes were created in a program called Terragen. The world was meant to feel extremely surreal, as the poem was set before time. To accomplish this effect, supernatural colors in the skyboxes were used by modifying the atmosphere and sunlight properties in Terragen. The cloudscape settings were changed to create large, ominous clouds, and after a satisfactory cloudscape was generated, the skyboxes were rendered. To properly render a skybox, 6 images are needed to make a cube: north, east, south, west, top, and bottom. The Terragen camera was set to specifically render each of these.

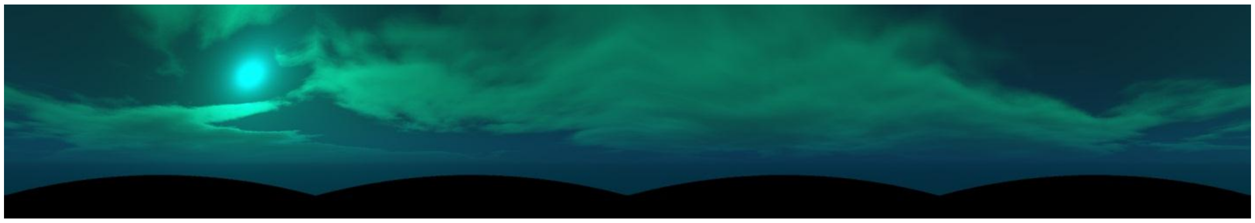


Figure 15. The "unfolded" skybox. The includes the renders from north, south, east, and west.

In Unity, the skyboxes were set up with a Skybox material. Each texture wrap mode had to be set to “clamp” in order for them to seamlessly come together and create the effect of a round, enveloping sky.

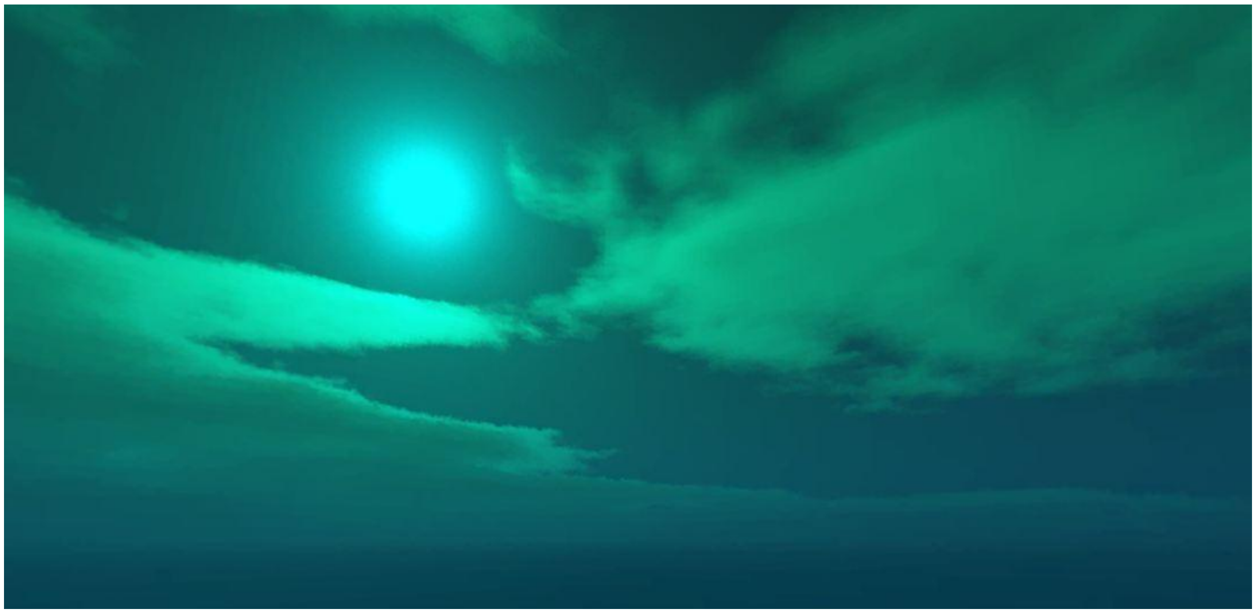


Figure 16. In game screenshot of the pre-earthquake Skybox

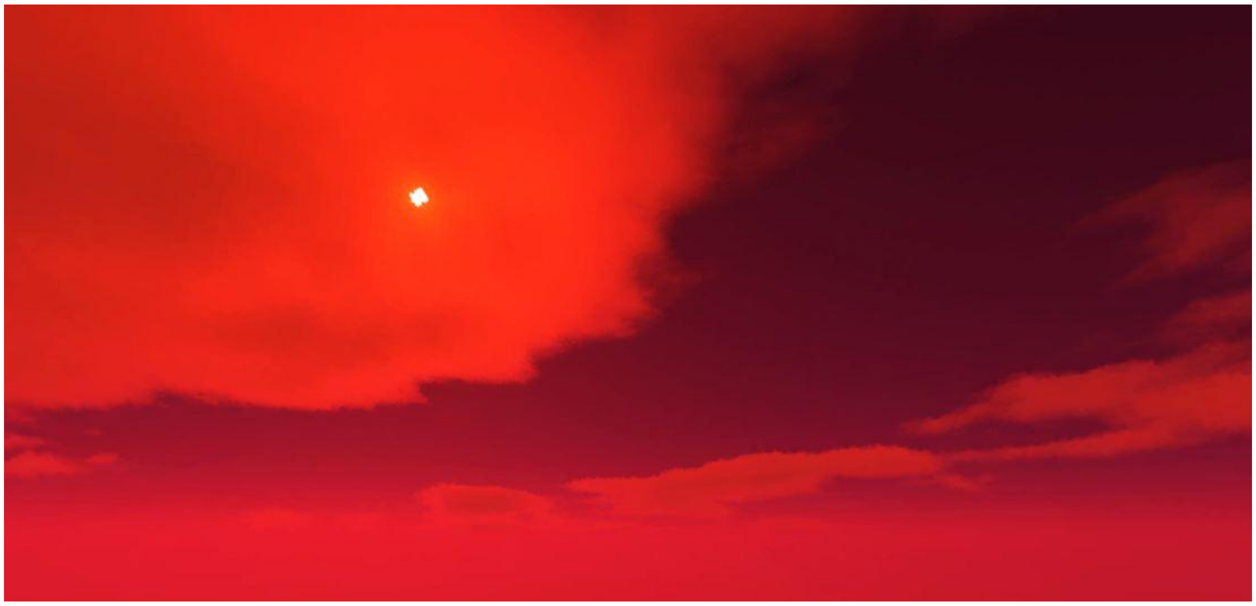


Figure 17. In game screenshot of the post-earthquake Skybox

Lighting the Scene

Lighting the scene was extremely important for setting the correct mood. The pre-earthquake scene was the first look the viewer would have at this strange world that was meant to be both peaceful and unsettling. At this point, the world was a blank slate. In line with this, a blue/green color pallet was chosen to emphasize a subdued and surreal world. With the skybox as a guideline, the pre-earthquake scene was filled with a dark green fog, a pale orange directional light, and a number of purple and blue point lights.

The post-earthquake scene was designed to be reminiscent of hellish landscapes, full of destruction. To contrast the initial scene, the post-earthquake world was filled with oranges and reds. To emphasize the colors chosen in the skybox, a deep purple fog and red directional light were used.

To correct for the excess amount of color in the scene, the fog was eventually de-saturated, but the bright skyboxes were left in order to give the trees and other objects a powerful silhouette. With this setup, lighting was much more effective and objects in the scene were properly illuminated.

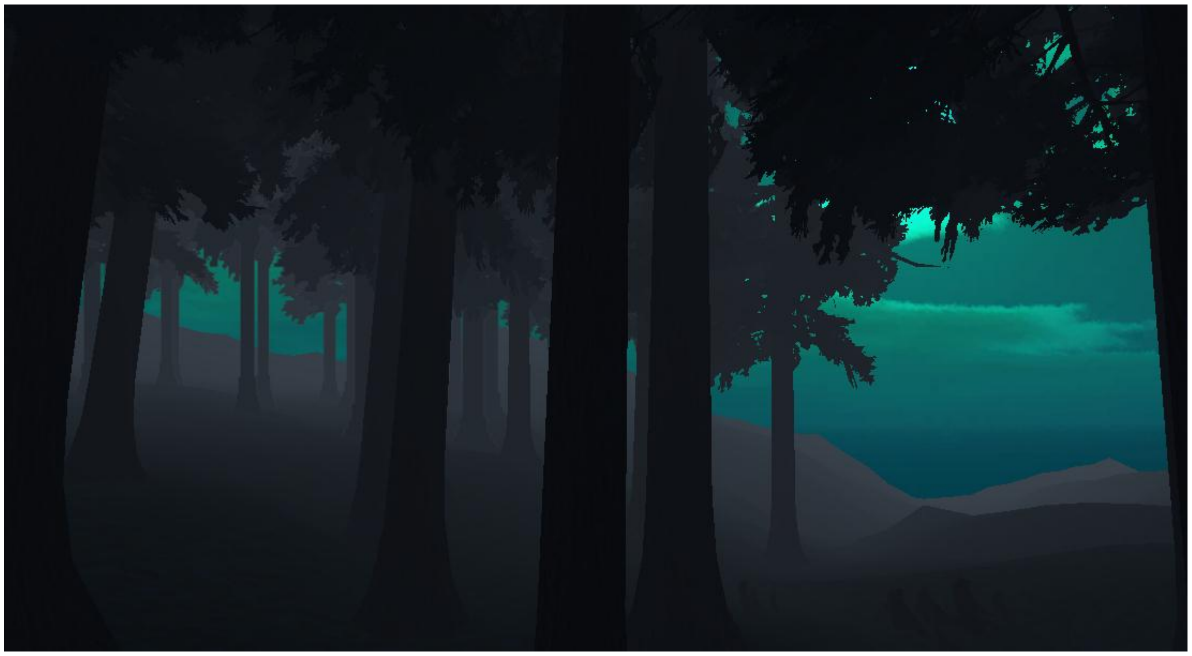


Figure 18 Desaturated scene

Using the Terrain Editor

The Unity terrain tool allowed a number of objects to be "painted" into a scene. These objects included brushes, grass, foliage, and textures.

Texturing

In realistic games, textures for the ground are often taken directly from a photo and modified slightly to be seamless. Because of Crow's surreal environment, the team wanted to create textures painted from scratch. To achieve this, photos were used as reference, but the textures were painted and abstracted from the original photo. The dirt became slightly too unusual, so to ground it, a photo sample was overlaid onto the image at low opacity.

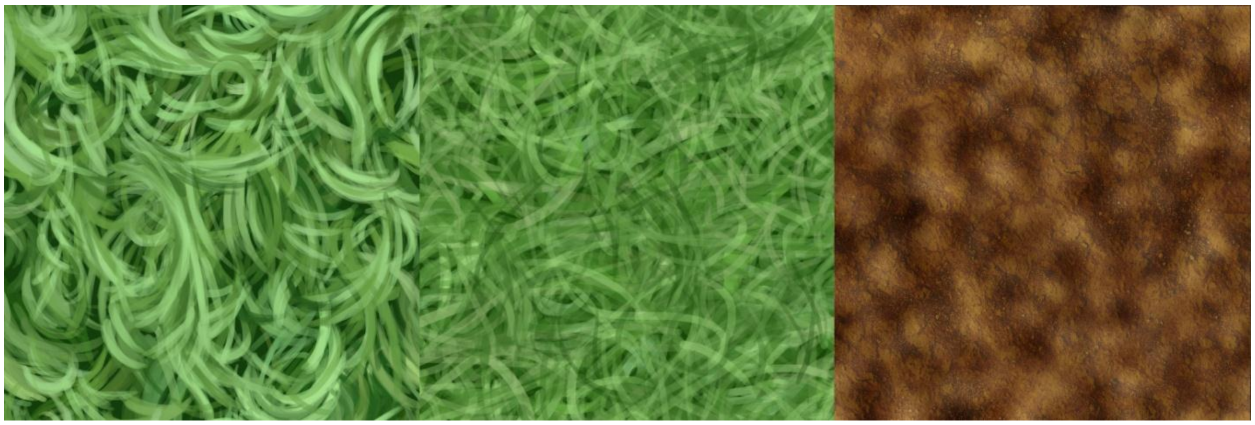


Figure 19. Some terrain textures. Left: 2 grass textures. Right: A dirt texture.

Trees and Foliage

The tree in the scene came from Terrain Assets[7], a royalty free pack of trees and foliage created specifically for Unity. It came with a number of free trees, but the one used in the scene was the Banyan tree.

[7] http://unity3d.com/support/resources/assets/terrain-assets.html

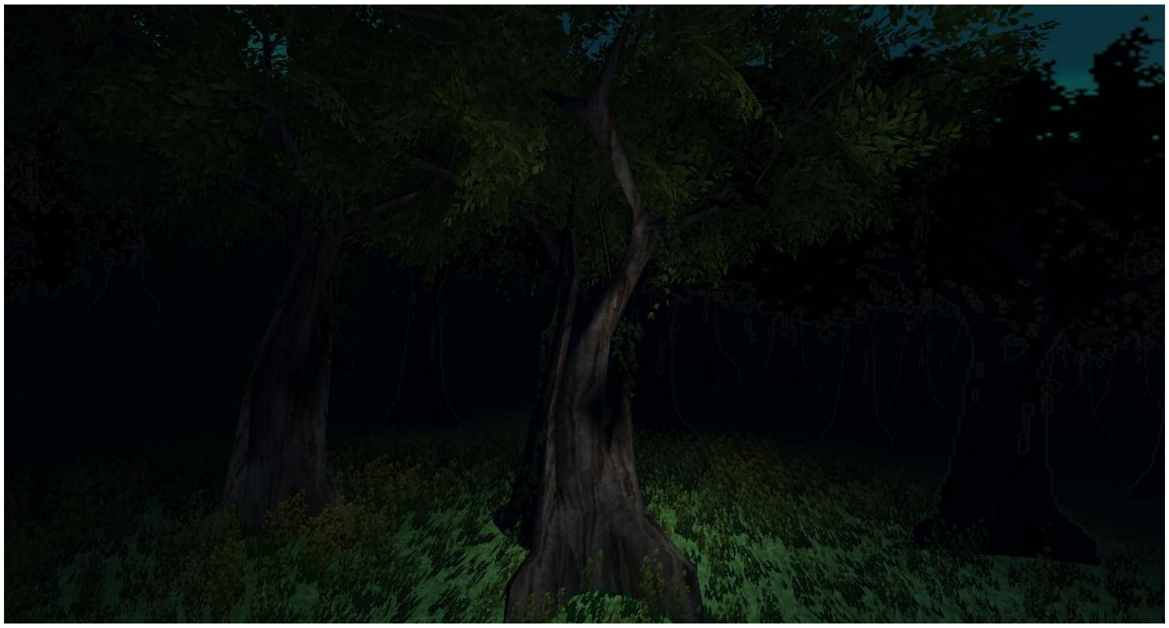
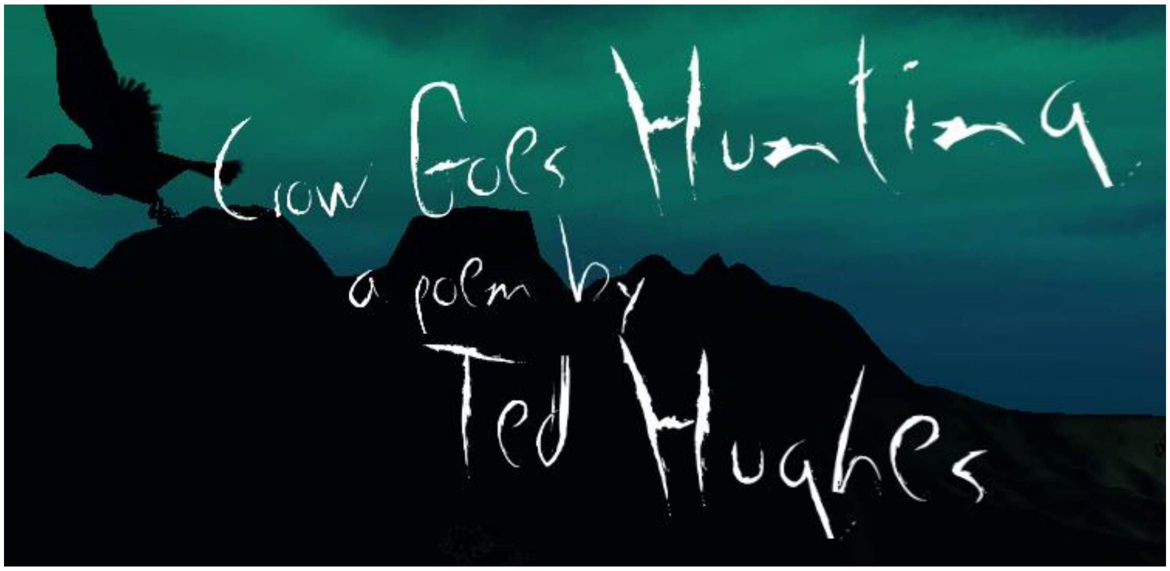


Figure 20 The final tree used in the scene.

3D Text

To display the poem's text, Unity's built-in "Text Mesh" was used. The tool created a 3D mesh with editable text. The words were then placed inside the scene, behind or above the action that the words described. An extremely organic font was chosen to help offset the harsh "CG" look that most of the scene carried.



Rain

A system called Rainscape[8] was used for the rain in the scene. It used a particle effect surrounding the camera for the downpour, as well as animated textures and particles for splashes and ripples on the ground.

Modeling Scene Objects

Rocks

The rocks in the scene were created to add interest to the environment. A low-poly base was created in 3DS Max, and imported into Zbrush with GoZ, a tool that transfers models between programs. The basic shape of each rock was individually sculpted in Zbrush. An interesting workflow was used to create the rocks at this stage; to quickly add detail; Zbrush's noise filter was used to flood each rock with small and large details. Crevices and cracks were then emphasized with the Damien Standard brush. Planar cuts were strategically placed in order to break up the noise and give the eye rest.

[8] http://u3d.as/content/ashnfara/rainscape/1Aj

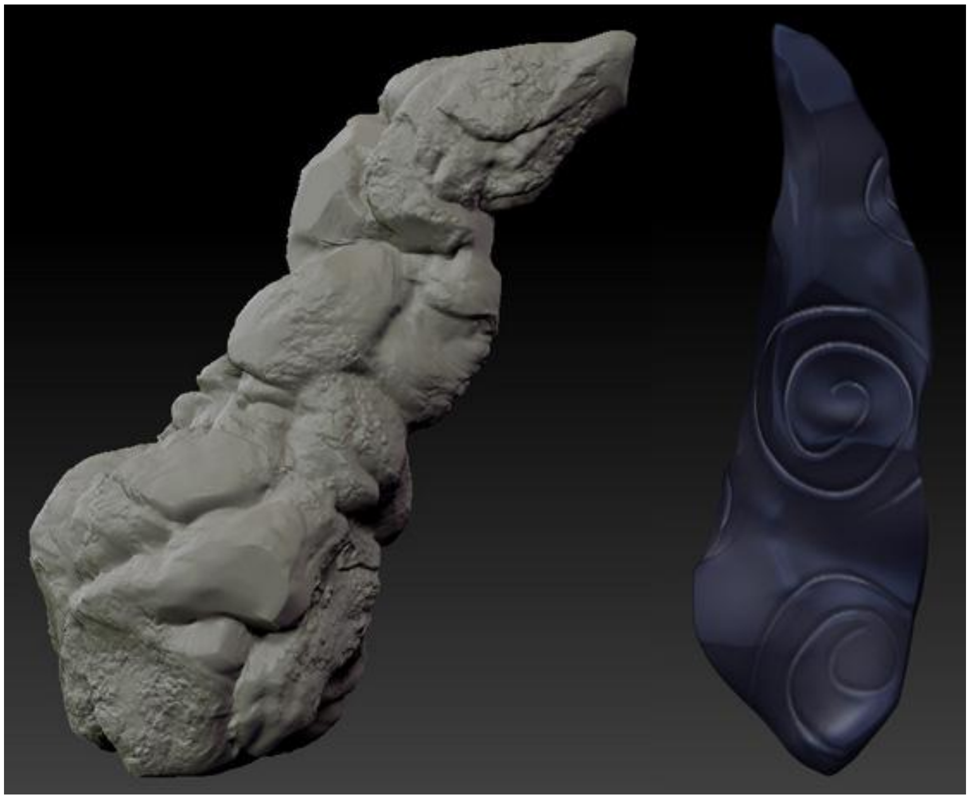


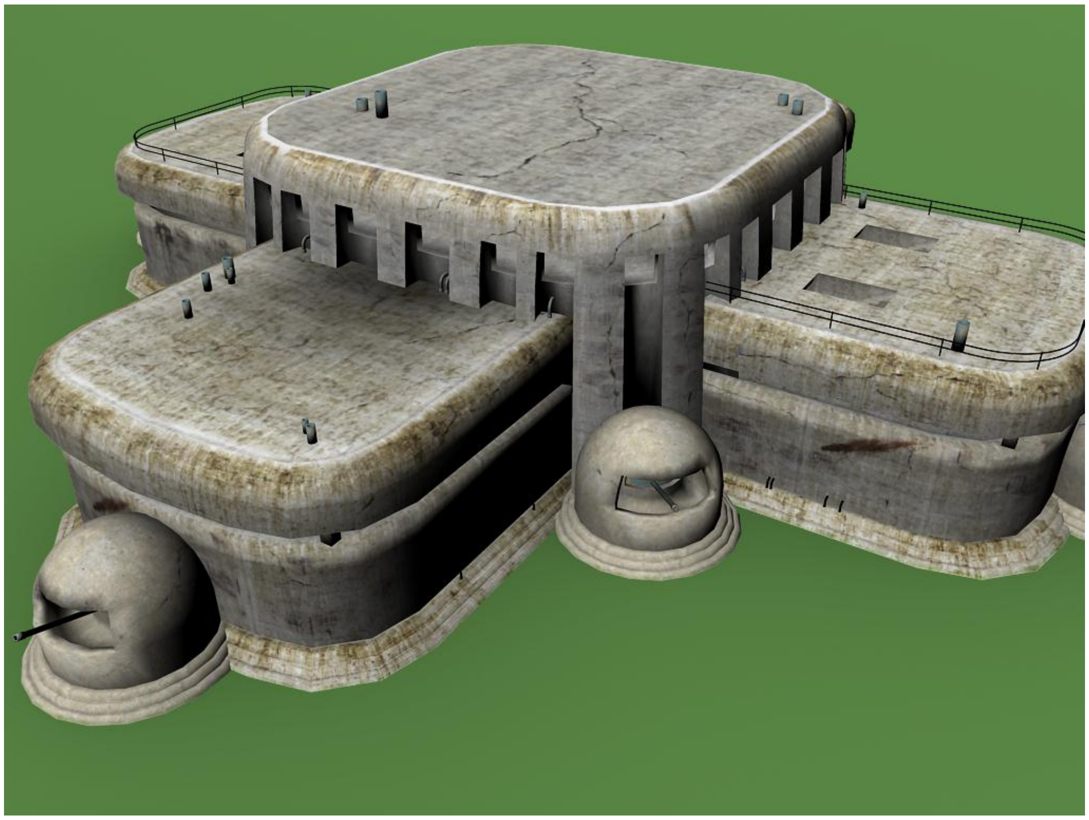
Figure 21. Zbrush screenshot of 2 rocks used in the scene.

To create the textures, the cavities were masked off and filled with dark colors. The cavities were then inverted and the rest of the rock was hand painted with Zbrush's Polypaint tool. Finally, Decimation Master was used to reduce the number of polygons to an amount that would function in our scene.

Bunker

The bunker was influenced by World War II style war bunkers and designed to appear damaged yet nearly impervious. It was modeled in Maya 2011 using several simple geometric shapes, and detail was added using bevels, inward and outward extrusions, face extraction, and other forms of mesh manipulation.

The bunker consisted of several different sections: one central square bunker, four side bunkers, eight cannons, and many small pipes and chimneys. One initial idea for the bunker was to animate the cannons firing at the lizard bombs, so the cannon was designed for the gun to swivel inside the cannon body.



War bunker with all types of sections visible.

Photoshop was used to create the bunker textures. The textures were sampled from a number of royalty-free images found online. CrazyBump was used to create the bunker's normal maps and Maya was used to render the ambient occlusion maps.

Reservoir

The reservoir was originally conceptualized as modern, square, and concrete. In order to better suit the Celtic theme, it was re-conceptualized with a more fantastical design.

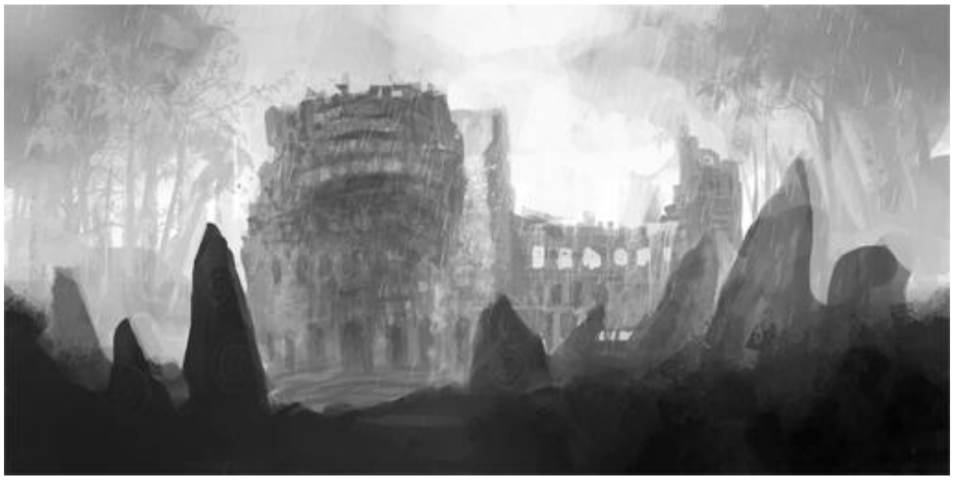


Figure 22 Reservoir Concept by Ryan Chadwick

The reservoir was modeled as a low-poly base in 3ds Max. The tunnels were created with Booleans, which caused problems when imported into Zbrush for high poly detailing. Despite this, stone alphas were used to create a detailed normal map, and Polypaint was used to create the colored texture.

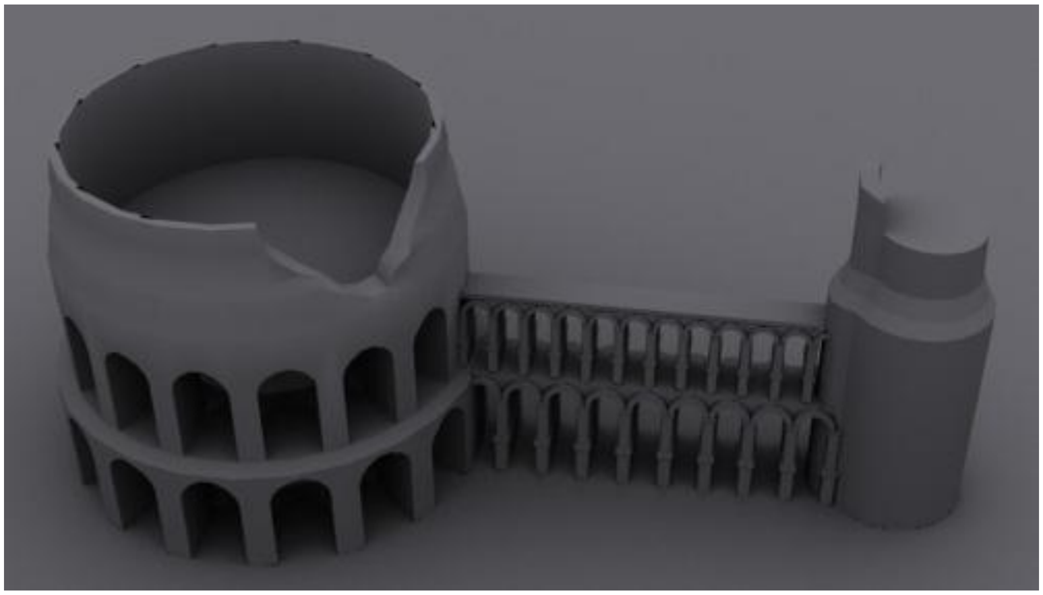


Figure 23 Low-poly Reservoir Model

Bomb

The bomb's initial, modern interpretation was referenced from a Paveway missile. It was created entirely in 3DS Max and the texture was hand-painted in Photoshop.

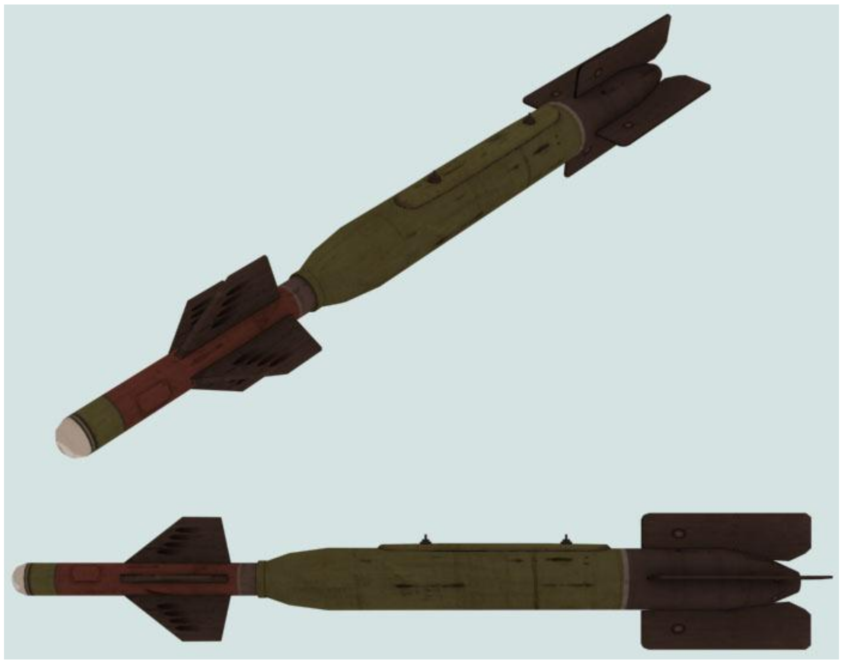


Figure 24 Initial model of the bomb

After the theme of the project was cemented, the bomb became fuming and reptilian. A source of inspiration sprung from Grond, the wolf battering ram of the *Lord of the Rings* series.

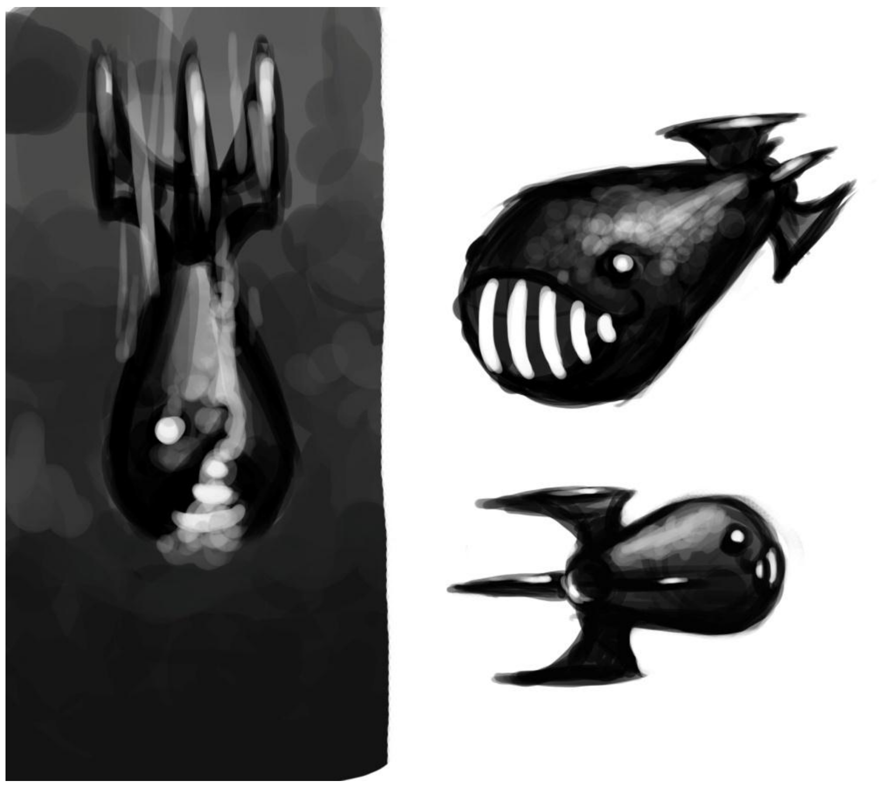


Figure 25. The reimagining of the bombs.

The basic structure was modeled in Maya and imported into ZBrush for sculpting. Enabling both X-axis and Z-axis symmetry allowed uniform manipulation of the fins and body, but Z-axis symmetry was later disabled to create the mouth-grate and face. The bones in the fins were a secondary idea, relating the object back to the shotgun lizard's fate – Crow needed the word for destruction, thus he didn't care what he broke along the way.

After retopologizing and UV mapping, the mesh was then Polypainted in ZBrush. The goal was to create a metallic look. A hint of soot around the openings was added for texture. The fins were colored to match the bone and skin pigments of the word lizards.

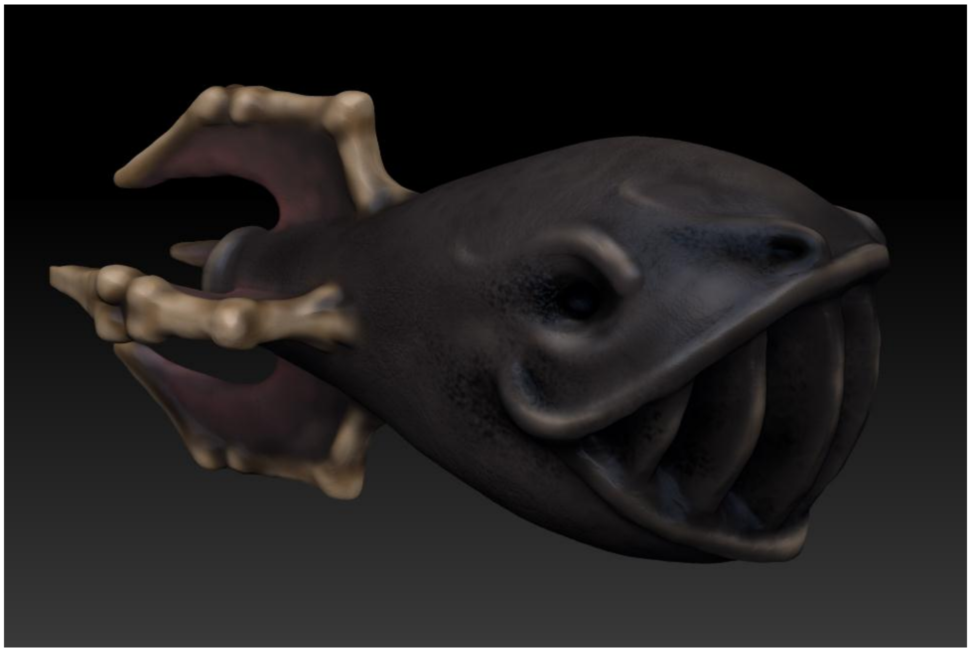


Figure 26. The final ZBrush bomb model.

To finalize the model in Unity, a fire particle effect was added, as well as 2 black spheres in place of eyes.

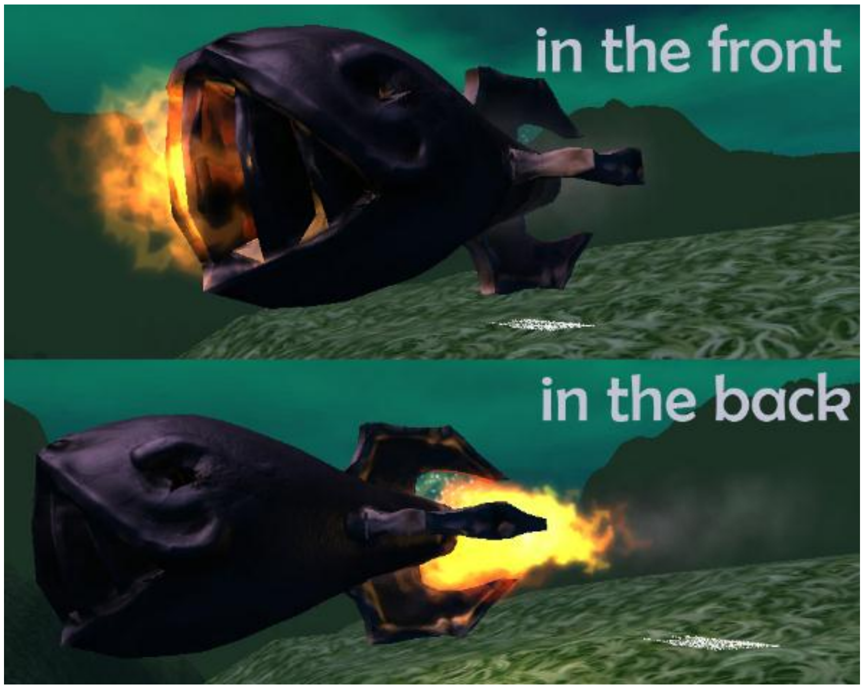


Figure 27 A test of the fire effects in Unity.

Shotgun

The original shotgun was modeled in 3DS Max and the texture was hand-painted in Photoshop, staying true to the modern form of a shotgun.

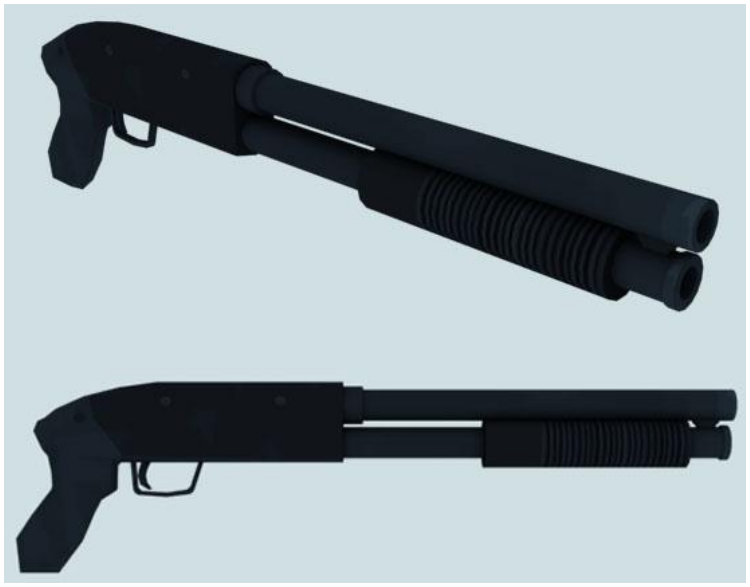


Figure 28 Initial model of the shotgun

However, when the theme was reimagined, concepts were drawn up to reflect the new ideals. Crow creates the shotgun from a word, and in light of the word concepts, it was clear that the shotgun should retain many of the original word lizard’s features. The gun became a word lizard taken in by Crow and formed to his will, with a shredded body and exposed innards.

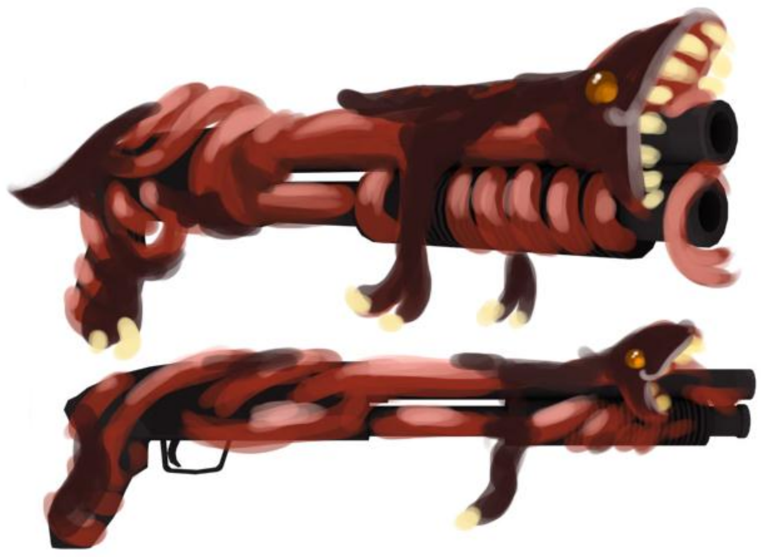
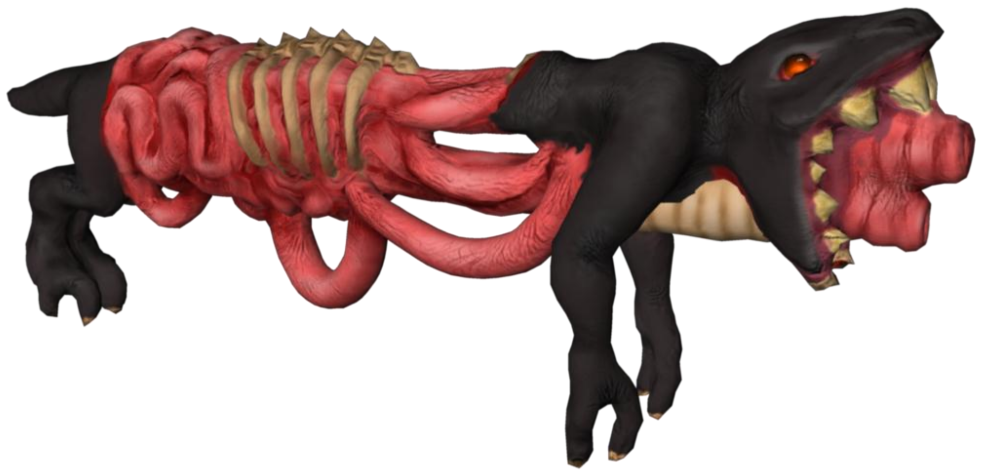


Figure 29. The reimagined shotgun concept. Note the lizard features prominent throughout.

The new gun model was created with the old guns base. A Zsphere body, entrails, and skeletal structure were created on top of it. While the skeletal structure and body were sculpted symmetrically, the entrails were asymmetrical. A gun-cocking mechanism subtool was also created to facilitate easy animation.

Details were added to the mesh, including internal organ wrinkles, skin bunching, and detailed vertebrae. Polypainted in Zbrush, the mesh underwent rigorous Mask Cavity use to allow intense detailing of the mesh's normals. The colors were taken straight from the word lizard's texture maps, and the entrails were given a diffused light and shadow treatment to emphasize the wet quality of the organs. To finish the piece, the cocking mechanism was reintroduced to the mesh, creating an entire gun bent to the will of Crow.



Rigging

Autodesk Maya 2011 was used to rig the models because it is a widely accepted program currently in use by professional game developers.

The first model rigged was the lizard. Initially, it was created with a large amount of joints to allow for a high degree of finesse in its motion. The rig was based around a “Center of Gravity” (COG) joint, which is a “root” joint, with all other joints as its children. Movement of the COG caused the other joints to move with it.

Unlike the COG and other joints, the legs were rigged with Inverse Kinematic (IK) handles. Pole vectors were added to the IK handles to allow control the direction that the middle joint – the knee – would point. IK Stickiness was also enabled, which caused IK joints to remain independent of other joints, such as the COG. This allowed for easy creation of a run cycle and other animations that involved precise leg motion. Additionally, every joint was named for the part of the body it controlled for easy identification in Maya’s outliner.

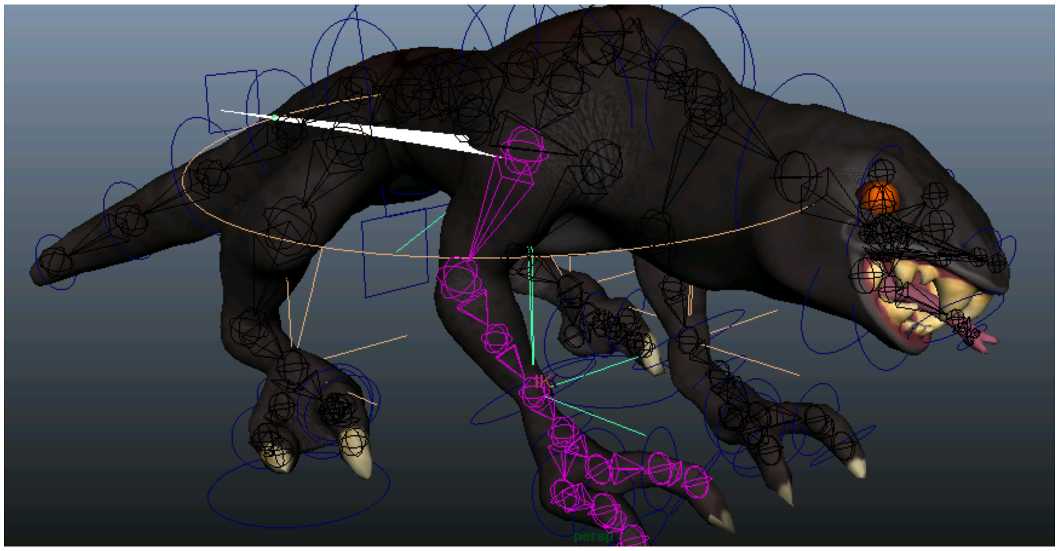


Figure 30IK handle (teal, with orange ring) controlling lizard rig's right arm joints (pink). Handle stretches from shoulder joint (indicated by white triangle) to ankle joint.

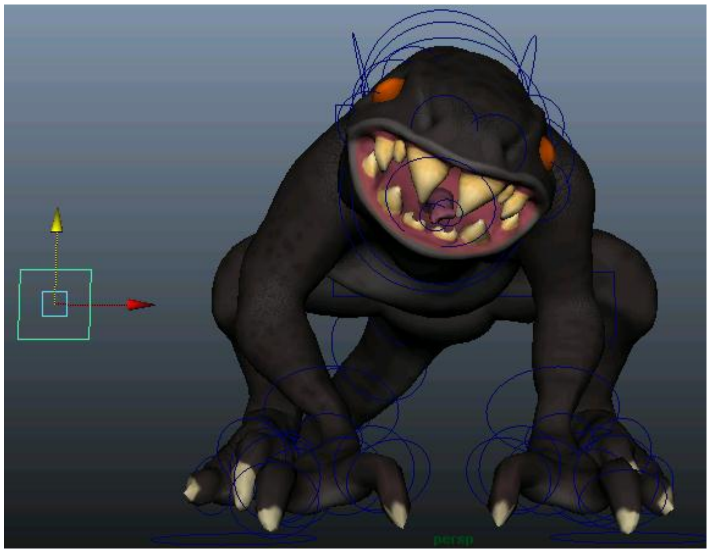
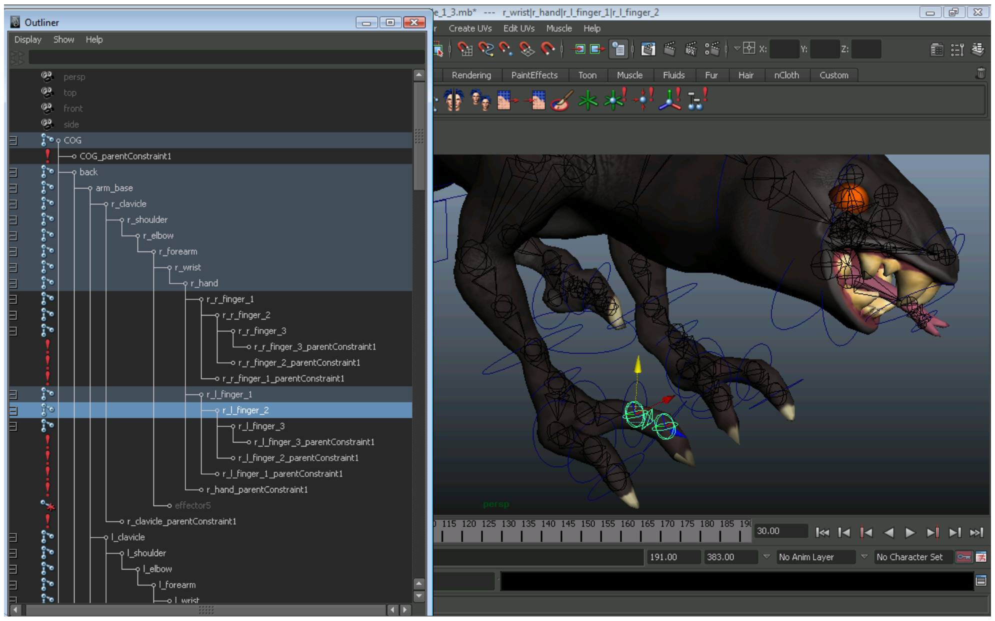


Figure 31 Changing the location of this pole vector changes the direction the elbow points to. The bending joint of a joint chain affected by IK will attempt to point in the direction of the pole vector.



```
Outliner
Display  Show  Help
persp
top
front
side
COG
└─ COG_parentConstraint1
└─ back
   └─ arm_base
      └─ r_clavicle
         └─ r_shoulder
            └─ r_elbow
               └─ r_forearm
                  └─ r_wrist
                     └─ r_hand
                        └─ r_r_finger_1
                           └─ r_r_finger_2
                              └─ r_r_finger_3
                                 └─ r_r_finger_3_parentConstraint1
                              └─ r_r_finger_2_parentConstraint1
                           └─ r_r_finger_1_parentConstraint1
                        └─ r_l_finger_1
                           └─ r_l_finger_2
                              └─ r_l_finger_3
                                 └─ r_l_finger_3_parentConstraint1
                              └─ r_l_finger_2_parentConstraint1
                           └─ r_l_finger_1_parentConstraint1
                        └─ r_hand_parentConstraint1
                  └─ effector5
         └─ r_clavicle_parentConstraint1
      └─ l_clavicle
         └─ l_shoulder
            └─ l_elbow
               └─ l_forearm
                  └─ l_wrist
```

Figure 32 Joints with consistent naming convention made it much easier to select correct joints in the Outliner. The "r_l_finger_2" (right hand, left finger, second from base) joint is selected.

Initially, the lizard rig was unnecessarily complex. The large amount of joints would be more time-consuming for Unity to compute, and the rig would not function optimally, especially when all five lizards were present onscreen. Therefore, the rig was optimized, primarily by removing joints along the spine and tail and in the face.

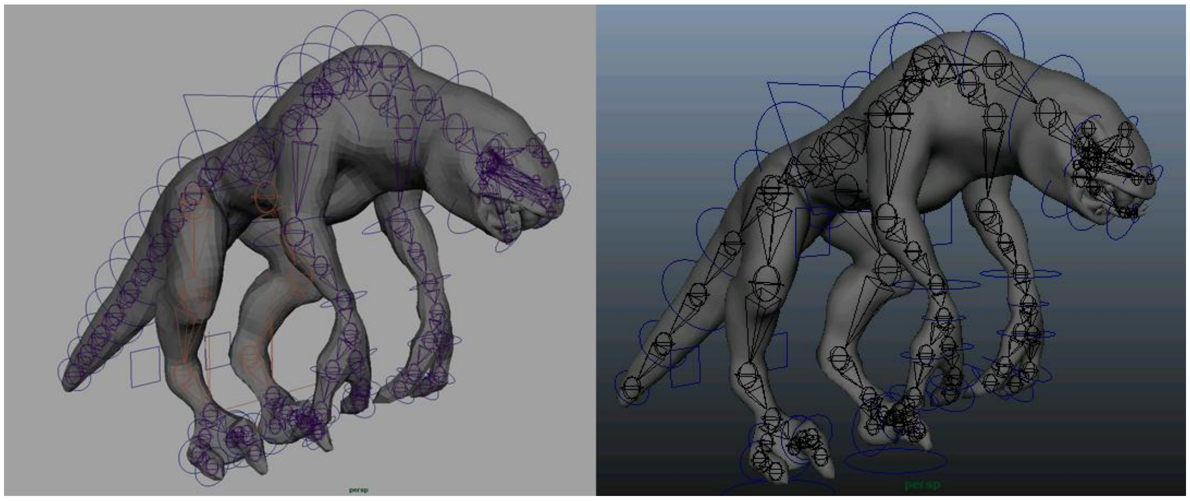
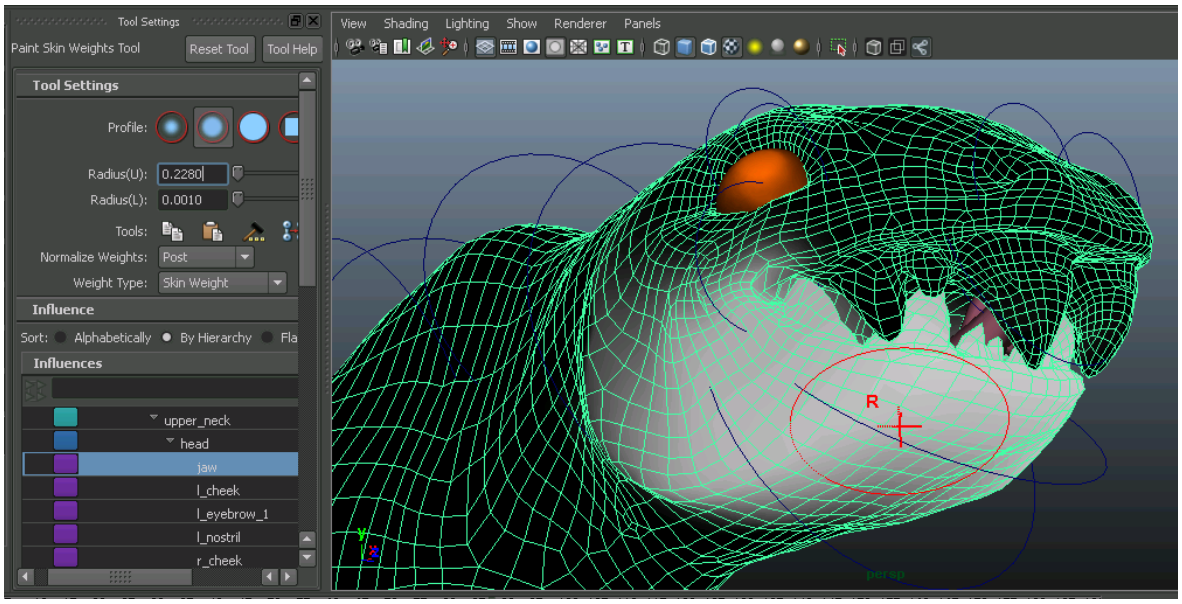


Figure 33. Left: the lizard rig before optimization. Right: The final lizard rig.

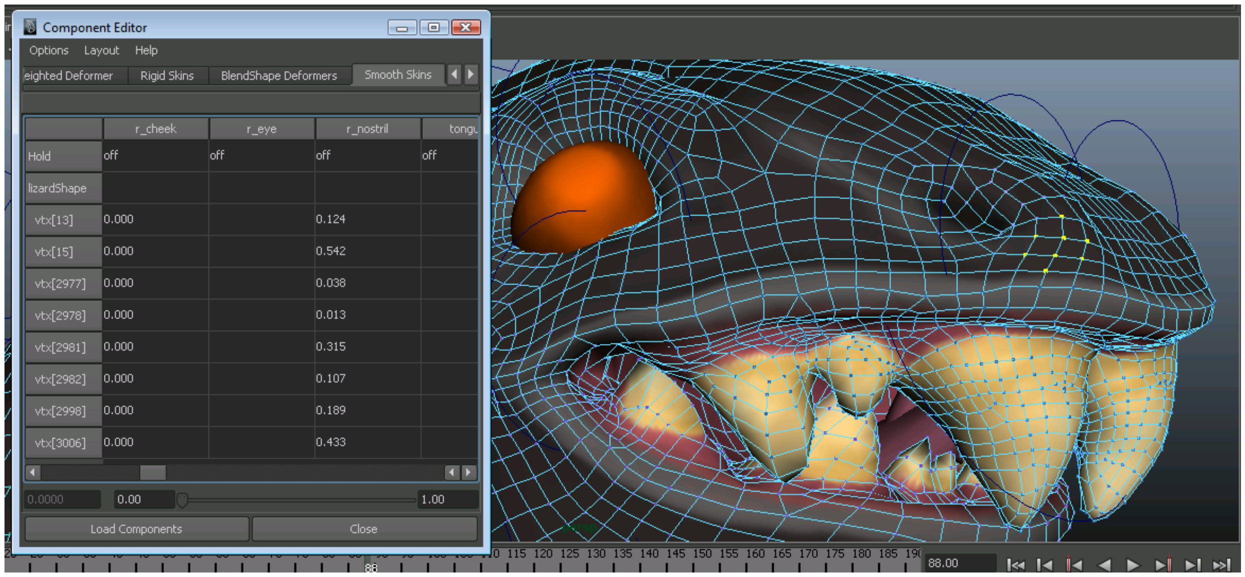
After setting up the joint system, a control curve was assigned to every joint. The joint controls were the blue circles surrounding or near each joint (or, in the case of the COG and pole vectors, a blue square). Like the joints, the joint controls were parented to each other as they approached the COG control. They were also given parent constraints to the joints; this caused the joints to move in the exact same manner as the controls. Controls were used on the joints because of the inability to freeze joint transformations (and the ability to freeze control transformations).

After binding the mesh to the joint system, Maya's skin weight painting tool was used to establish how the mesh's vertices were influenced by the joints. Maya's Component Editor was used to edit the numeric value for vertices near particular joints for individual vertices.


Paint Skin Weights Tool

Radius(U): 0.2280
Radius(L): 0.0010
Normalize Weights: Post
Weight Type: Skin Weight
Sort: Alphabetically ● By Hierarchy Fla

Figure 34 Maya 2011 Paint Skin Weights Tool interface with lizard jaw joint selected. The whitened area on the mesh displays the vertices affected by the joint. The whiter the area directly around a vertex, the more influence that joint has on that vertex.



	r_cheek	r_eye	r_nostril	tongu
Hold	off	off	off	off
lizardShape				
vtx[13]	0.000		0.124	
vtx[15]	0.000		0.542	
vtx[2977]	0.000		0.038	
vtx[2978]	0.000		0.013	
vtx[2981]	0.000		0.315	
vtx[2982]	0.000		0.107	
vtx[2998]	0.000		0.189	
vtx[3006]	0.000		0.433	

0.0000 0.00 1.00

Figure 35 The smooth skins menu. The selected vertices were each assigned a particular influence value for the "r_nostril" joint. The sun of influences values for any vertex across all joints will always equal 1.

The hare rig was created in the same manner as the lizard; with a focus on optimization. Joints were included in the ears to animate realistic, floppy ear movement, and joints were also added in the tail, nose, and cheeks.

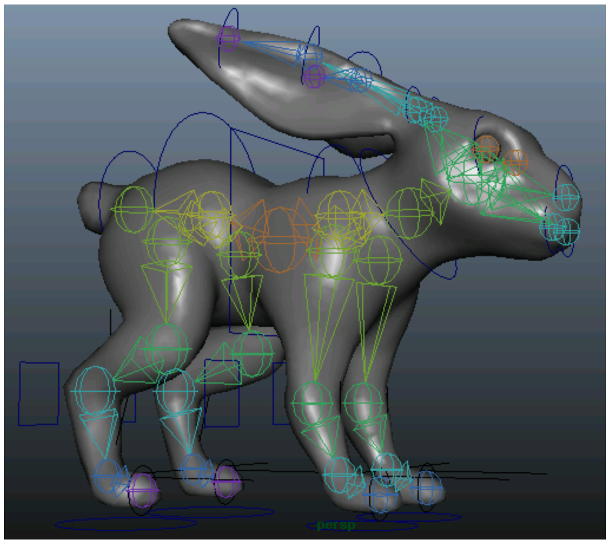


Figure 36 Hare rig

There were initial problems getting the lizard and hare rigs to animate correctly in Unity. It was discovered that animations needed to be baked from the joints to the mesh itself in Maya rather than with Unity's automatic baking. All the joints and applicable transformation attributes (translation and rotation) were selected, and the keyframes were baked onto the mesh. Afterwards, the joints and IK handles were deleted, and the file was optimized. The scene became an animated mesh with no rig; it was exported as an FBX and worked perfectly in Unity.

The crow rig was initially built with a rigging tool that was later abandoned in favor of manual rigging. This rig was the most complex of all the rigs because of the crow's role as the main character of the story. The joint system was designed to be very similar to a real crow, particularly in the wings and legs.   There was also a joint for each of the crow's six eyes. The skin weight painting for each eye joint included the area directly surrounding the eye; this allowed the skin around each eye to move as the eye moves, therefore emphasizing the crow's wild and freakish eye movement.

The preliminary work with the rigging tool left unwanted information in the rig file, which caused problems when importing the crow into Unity. In Unity, the crow mesh was extremely distorted; the body and wings disappeared, leaving only the head, which was deformed. After analyzing the crow rig file and conducting outside research, it was determined that the rig had two problems.

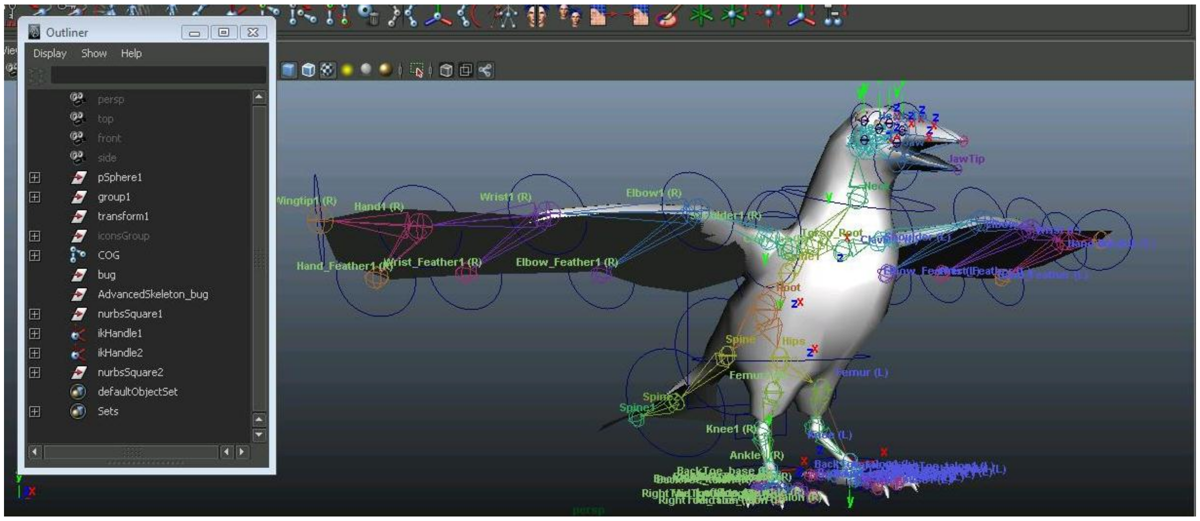


Figure 37 Crow rig. This was distinctly different from the lizard and hare rigs, and it contained unwanted information as reflected in the Outliner (such as “transform1,” “iconsGroup,” “bug,” and “AdvancedSkeleton_bug”).

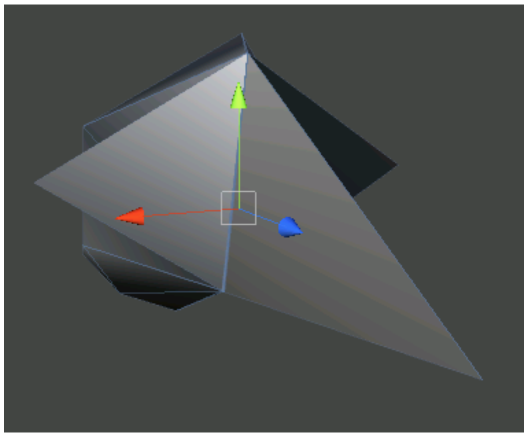


Figure 38 When imported into Unity, the Crow became unrecognizable.

The first problem was that a joint in the beak contained information that was causing malfunction, and it was not clear how to delete the information without deleting the joint. A Maya error message declared that the information, a transform, was non-orthogonal and therefore unsupported by the conversion of the crow file to an FBX - a widely accepted file format for game animations. The second problem was that the crow rig contained multiple bind poses, and Unity accepts meshes with only one bind pose.

A Maya MEL script written by Elliot Borenstein was used to delete the unwanted beak joint information, and a Python script was used to eliminate multiple bind poses. The only negative side effect to applying the MEL script was the need to repaint the skin weights in the beak area. Despite the use of the scripts, Unity would not accept a crow animation file if it was converted to an FBX in Maya. However, if the Maya file was imported into Unity and automatically converted, it worked fine.

Animation

Maya 2011 was used to animate all the models. There were two types of animations: looping and transitional. Looping animations occurred during scenes, and transitional animations occurred during the transition time between scenes.

The animations were intended to capture the emotions of “Crow Goes Hunting.” The crow needed to be cunning and twisted, the lizards had to be feral and aggressive, and the hare possessed intelligence, quick wit, and benevolence. While blocking in each animation, the type of attitude we wanted each character to exhibit was emphasized in their movements.

A full list of animations, including those that were omitted from the final build, is available in the appendix. A list of descriptions of each animation follows.

Crow

Flight loop

This looping animation occurred during the title screen. The crow is seen flying ominously over the landscape, flapping its wings occasionally. Reference on bird anatomy and behavior was used to animate this accurately.

Landing

This transitional animation occurred as the crow landed on the rock to summon the lizards from the ground. The crow slowly touched down on the ground while surveying the area carefully. The aforementioned crow reference was also used to animate this accurately.

Pointing

This looping animation occurred when the crow pointed out the hare to the lizards. The crow beckoned the lizards by waving its right wing towards its position, then taking a heavy breath and pointing its left wing at the hare while verbally exclaiming. This animation solidified the crow's position as the leader of the lizards.

Summoning

Combined with the landing animation, this transitional animation occurred when the crow lands and summons the lizards from the ground. The crow bows then raises and spreads its wings above its head twice. This animation displayed the crow's powerful, supernatural capabilities.

Lizard

Emerging

This transitional animation occurred after the crow summons the lizards from the ground. The lizard trusted one arm out of the ground, followed by its other arm, then raised itself up. It roared ferociously as emerged, and kicked through the dirt to bring itself up to ground level. This animation presented the lizards' savage, violent behavior.

Idle

A unique, looping idle animation was created for each of the five lizards. These were combined with the emerging animation and occur directly afterwards. The lizards performed various actions, such as standing on hind legs, scratching, punching the ground, sniffing the air, or roaring. These animations reinforced the lizards' personalities and offered variety among the lizards' behaviors.

Run cycle

This looping animation occurred when the lizards chased the hare. The run style is similar to that of many real lizards, primarily the Komodo dragon. Emphasis was placed on slithering tail motion and ravenous demeanor.

Hare

Run cycle

This looping animation occurred when the hare is being chased by the lizards. The run style is similar to that of real life rabbits and hares. Reference of the details of a rabbit run cycle, provided by Meagan Boucher, was used to animate this accurately. This animation emphasized the hare's agility and grace.

Idle

This looping animation occurred after the crow pointed out the hare to the lizards. The hare innocently sniffs the ground and occasionally glances up as if it heard a disturbance. This animation portrays the hare's innocence, curiosity, and awareness.

Tech

The project was created in the Unity game engine, an engine popular for its gentle learning curve, versatile capabilities, and ability to port projects to the web, a major goal for the team.

Node-Based Camera System

To create the camera movements, a node-based camera system was used, utilizing a script written by WPI Student Eric Walston. The nodes consisted of two parts: a CameraNode and a LookAt point. The CameraNode was the point that the camera would move to, and the look at point was the spot that the camera would be pointing to. The look-at node was connected to its respective camera node, and the camera node was placed into the main camera, so that it would move from point to point.

Each node also had a transition option; tween, cut, or GoThrough. Tweens would take a specified number of seconds to transition from one node to the next whereas cuts were instantaneous transitions. GoThrough nodes were slightly more complicated in that they were used as additional points on a curve that would not be stopped at, to make a tween transition smoother, and also to help trigger objects invisibly behind the scenes. GoThrough nodes were used to trigger animations and help transition from one animation to the next.

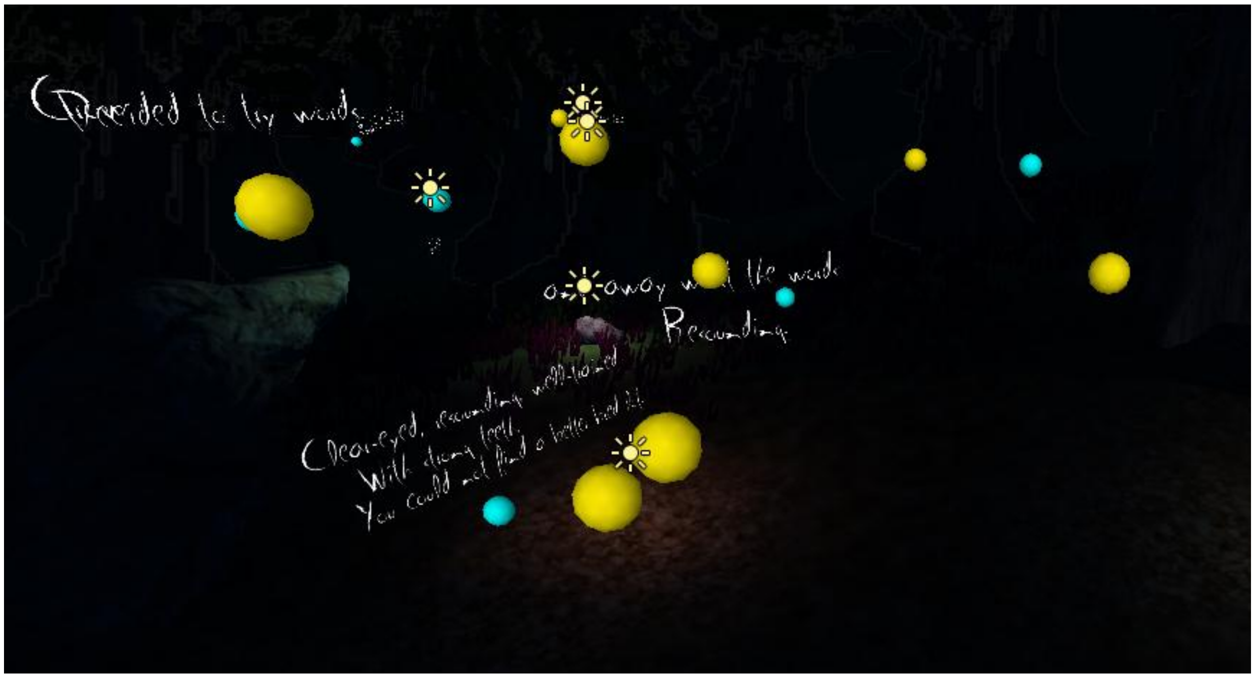


Figure 39 A screenshot of some of the nodes used in t he scene. The Yellow were CameraNodes, and the blue were LookAt points.

Animation Pathing

To move objects in the scene, a system very similar to the CameraNodes was implemented. A breadcrumb trail of nodes was set up in a loop for objects to follow. The nodes were assigned to the objects that needed to move on that path. The system was somewhat limited, but it served its intended purpose.

Post-Mortem

What Went Right

Overall, the project was an extremely positive experience. The team's main goals were to be challenged and become stronger in their respective fields, and this was true across the board. Every member learned a multitude of new techniques and grew as an artist. Most importantly, with only a few exceptions, all the models, animations, and textures initially planned to be created were successfully made and imported into the game engine. This is a huge success because the team was able to create a world from scratch that looked great.

The team was also extremely self-motivated and good at staying on their toes, especially considering the timing of the MQP and the scattered location of the team members. The team had to create everything from scratch, and solve almost every problem on their own. The project was constantly changing to suit the needs and abilities of the team, which required strong leadership to keep under control. There was also a strict schedule and constant planning and milestone development was a huge factor in keeping the team cohesive and working well. Lacking a traditional "meet every week" type of schedule meant that the team had to be extremely reliable and have excellent communication, so they met every day online for extensive problem solving and work sessions.

What Went Wrong

The biggest roadblock the team encountered was a lack of a programmer. Not having a technical member made some of the more complex ideas the team had impossible. This also caused a huge time sink because team members had to constantly step out of their artist roles and attempt to troubleshoot and understand technical aspects of the project. There were also a number of things we had hoped to achieve but were impossible without a programmer.

Another big problem with the project was scope. For a small group, working on such a large poem was a mistake. Despite working constantly, the team was unable to fully implement the poem, thus rendering a number of models and animations useless. In hindsight, utilizing Aesop's Fables as our 3d narrative may have been the wiser choice. They were initially rejected for their simplicity; clearly that simplicity was something that should have been sought out. Additionally, the options for re-conceptualizing the fables were enormous yet manageable. Creating a simple world with a high level of polish would have been much more desirable than a half-completed project.

Another issue the team struggled with was arriving at the final idea of creating *Crow Goes Hunting*. The development process went through many stages. Initially, it was a continuation of an IQP called *Abyme,* which was a 3D sidescroller created in the Unreal engine. It was meant to be a game with a message, but the inability to find a programmer led to scrapping the idea and starting from scratch. The first non-programmer idea was creating a board game with game pieces modeled in Zbrush and 3D printed. This idea did not make sense in the context of our project because it didn't allow the animator or environment modeler any chance to improve on their craft. Planning started over a third time, but this time the team laid out clear goals to guide our process. This was much more successful and led to the current iteration of the idea: creating a story and translating it to a 3D, web-based medium. The story options were distilled into three options: the creation of an original work, Aesop's fables, or a poem. The team decided on a poem, specifically *Crow Goes Hunting*, and the project work could finally begin.

Goals for Possible Future Implementation

Time constraints prevented us from achieving all of our project goals. Therefore, our first goal for a future, final iteration of our project would be to complete our representation of the poem by implementing all of the assets and animations created for the project. We would also refine our creations by adding finer detail to our models and textures to achieve higher visual quality, and would polish our animations to attain greater quality and believability.

A major factor missing from this project was audio. Without sound, many animations and situations lack both context and tension, which is key in a dramatic retelling of any story. Had we more time, sound effects and audio would be added to enhance the mood and setting of the poem.

Additionally, the current system for animation transitions is rather convoluted. With the aid of a programmer, we would make it possible for one animation to automatically crossfade into the next without the need for exactly timed GoThrough nodes and complicated scripting.

Water shaders are extremely diffuclt to master, and there aren't an extraordinary amount of resources available online. In the future, we would like to enlist the help of a technical artist to create more realistic water effects for the scenes with rain and the reservoir.

Finally, near the end of the project, an alternate method of displaying the poem was considered. The idea was replacing the binding camera system with the ability to walk around the scene. The viewer was guided with marked arrows on the terrain as well as invisible walls as they travelled through the story. The benefit of this approach was clear: it introduced a new layer of interactivity and interest. The ability to explore a scene is much more engaging than simply clicking through predetermined camera shots.

Works Cited

Borenstein, Elliot. "Python Script for Crow Rig." Newton, July 2011.

Hughes, Ted. "Crow Goes Hunting." Crow. Faber Paperbacks, 1974. 48.

Walston, Eric. "Unity Camera Script." Worcester, June 2011.

Appendix

Crow Goes Hunting

By Ted Hughes

Crow

Decided to try words.

He imagined some words for the job, a lovely pack-

Clear-eyed, resounding, well-trained,

With strong teeth.

You could not find a better bred lot.

He pointed out the hare and away went the words

Resounding.

Crow was Crow without fail, but what is a hare?

It converted itself to a concrete bunker.

The words circled protesting, resounding.

Crow turned the words into bombs-they blasted the bunker.

The bits of bunker flew up-a flock of starlings.

Crow turned the words into shotguns, they shot down the starlings.

The falling starlings turned to a cloudburst.

Crow turned the words into a reservoir, collecting the water.

The water turned into an earthquake, swallowing the reservoir.

The earthquake turned into a hare and leaped for the hill

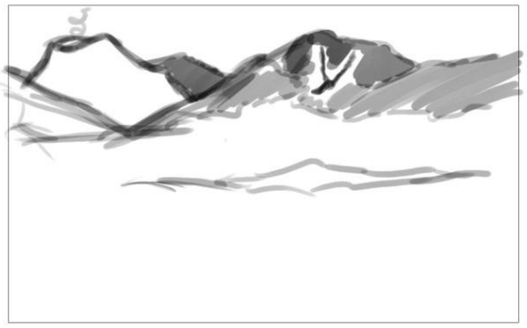
Having eaten Crow's words.

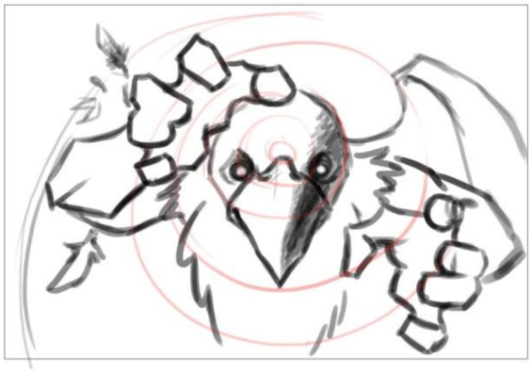
Crow gazed after the bounding hare

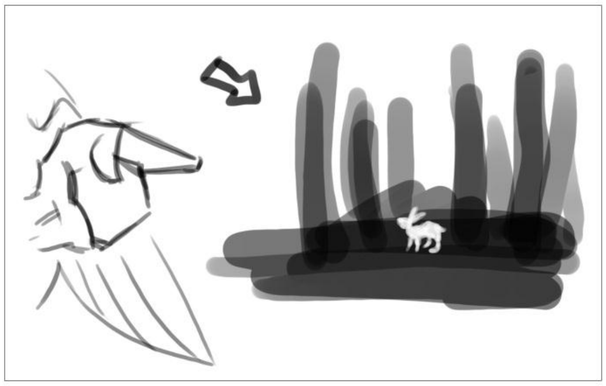
Speechless with admiration.

Storyboard

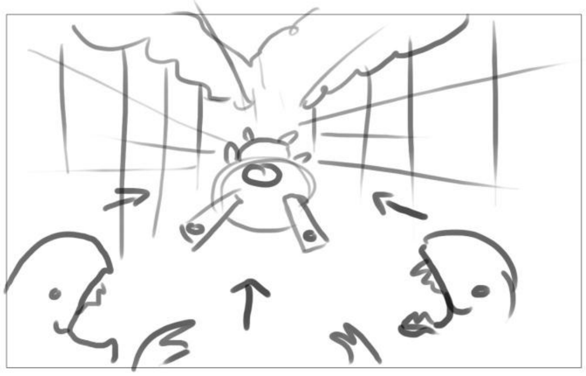
CROW
Goes hunting







ACRID
HATE
Haben perspective.

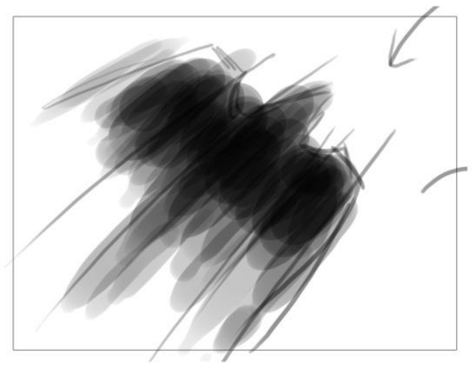
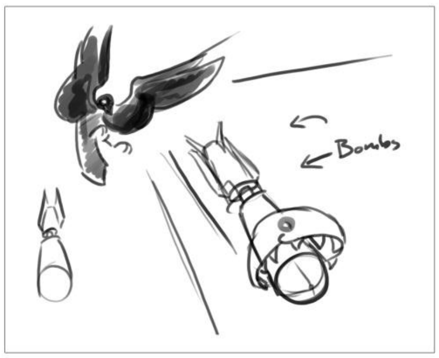


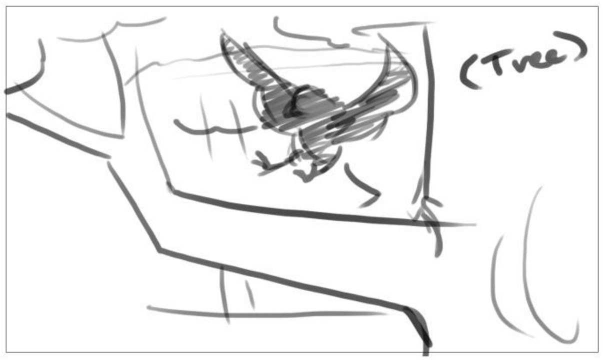
(rabbit hole)
snarles that way

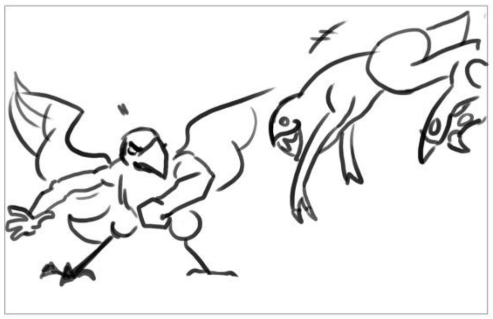
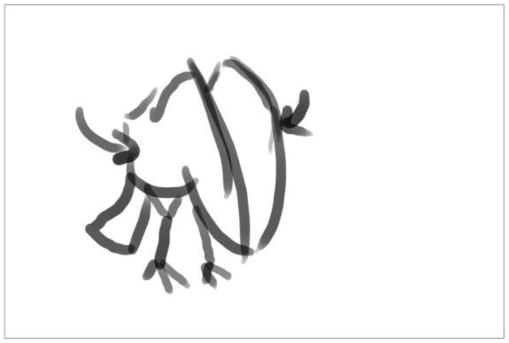
Rumble!

Distills smaller birds
WHUMP!

OR
(on the bunker)



Bombs

BOOM.

(Tree)

(follow)
Bob Ross Trees.
RUMBLE
DK
(Destruction)

Bunny run's away

scuse me

...
Bumbles
FIN

Additional Concepts

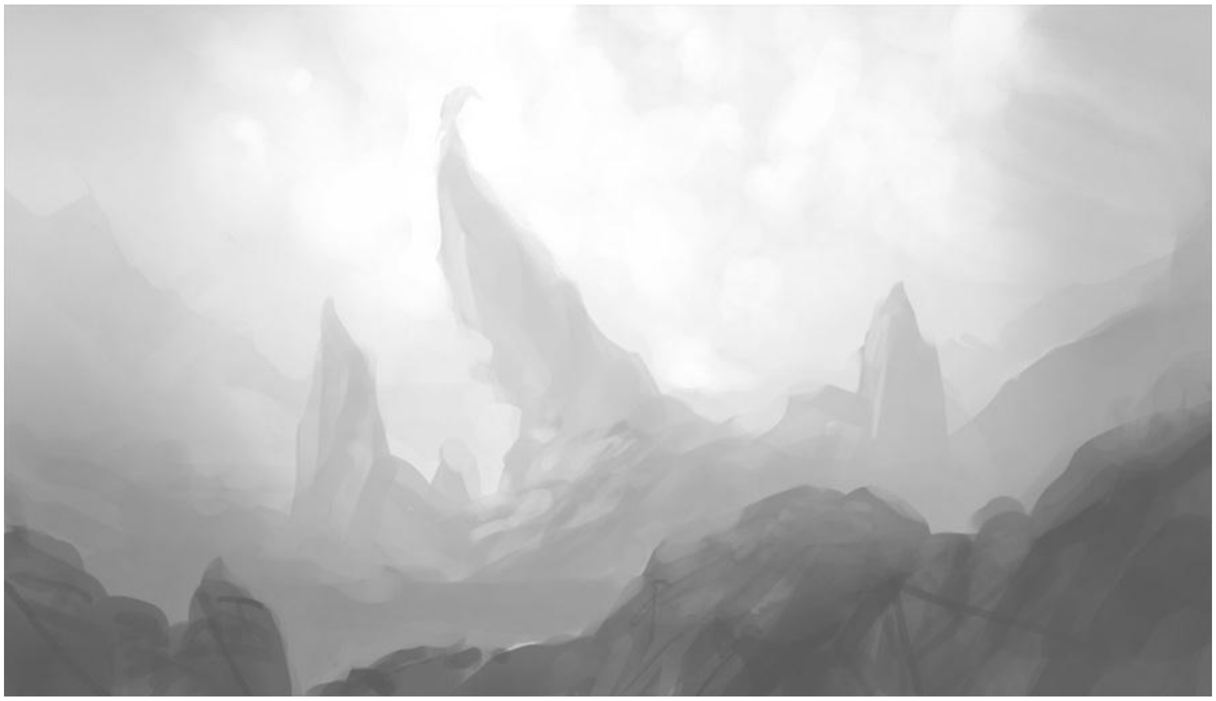


Figure 40 Environment Concept by Ryan Chadwick

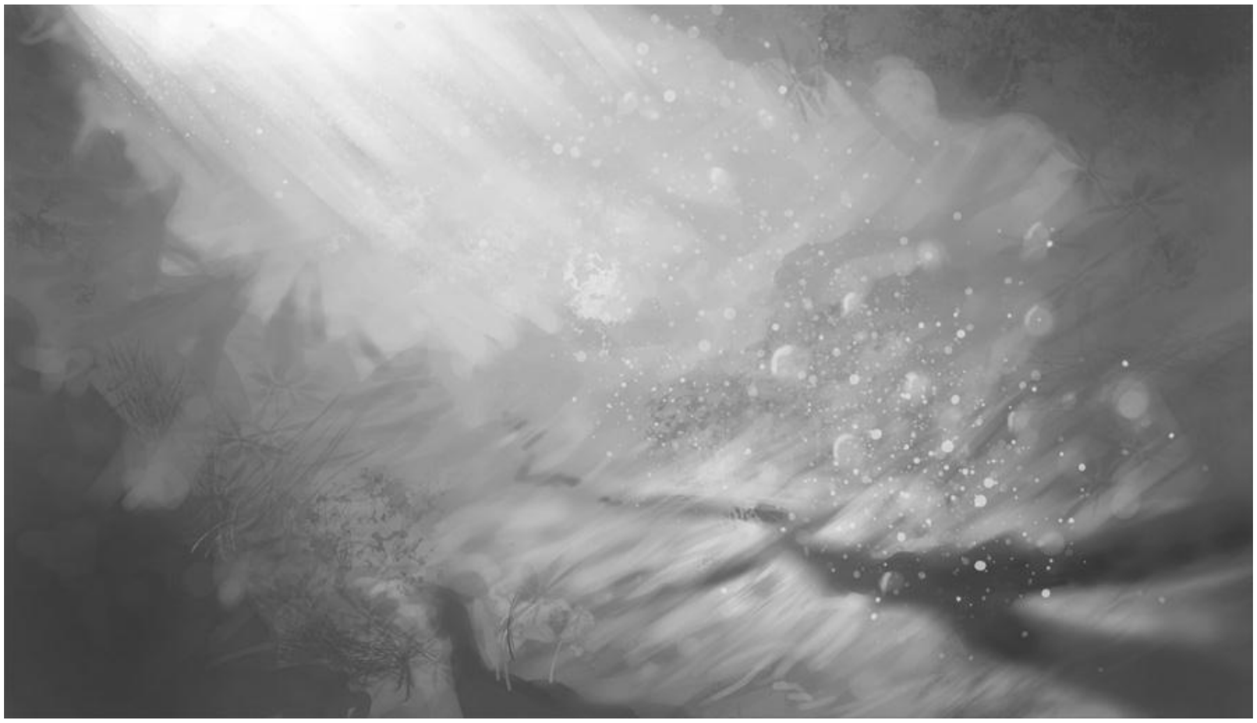


Figure 41 Environment Concept by Ryan Chadwick

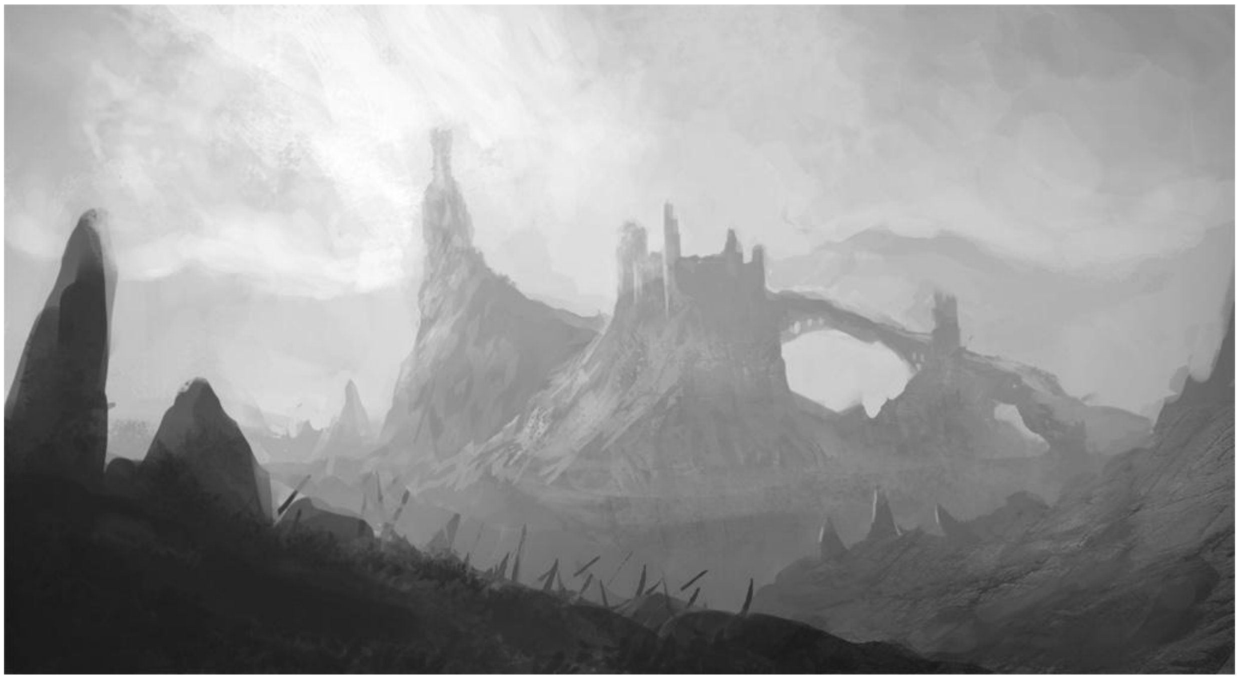


Figure 42 Bunker Concept by Ryan Chadwick

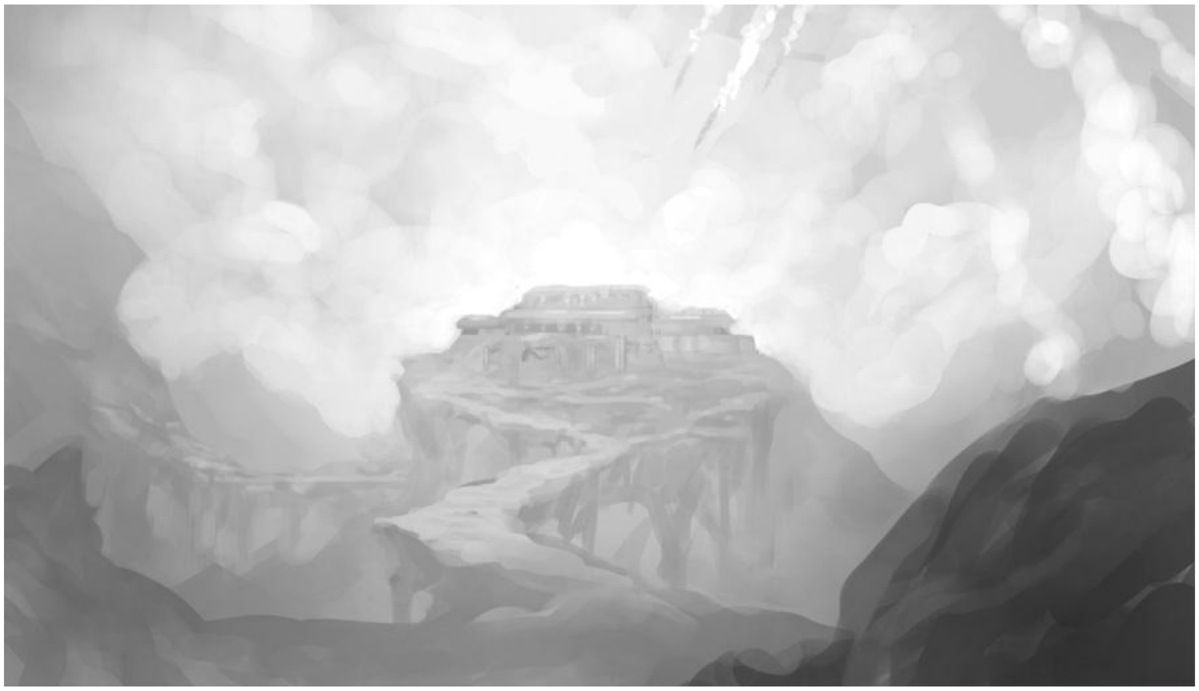


Figure 43 Bunker Concept by Ryan Chadwick

Asset List

This is the final asset list. Items that were completed are marked with a C. Items that were completed and were able to be implemented into the game are marked with a C+.

Assets	Status
Models and Textures	
Crow	C+
Lizards	C+
Hare	C+
Starling	C
Starling Particle Effect	C+
Shotgun	C
Bomb	C+
Reservoir	C+
Bunker	C+
Terrain	C+
Trees	C
Rabbit hole	C
Grass Texture	C+
Dirt Texture	C+
Rock texture	C
Ground rock	C

Set Piece rock	C+
Carved rock	C+
Foliage	C
Billboarding Grass Texture	C+
Firefly particle effect	C+
Rigging and Skin Weighting	
Crow	C+
Lizards	C+
Hare	C+
Starling	C
Animation	
Crow flight loop	C+
Crow glide	C+
Crow taking off	C+
Crow landing	C+
Crow pointing	C+
Crow releasing bomb	C
Crow idle	C+
Crow firing shotgun	C+
Crow gazing	C+
Crow petting lizard	C

Crow swooping	C
Crow feeling rain	C+
Crow taking in lizard	C
Crow summoning	C+
Lizard emerging	C+
Lizard idle (x5)	C+
Lizard run cycle	C+
Lizard being pet by Crow	C
Hare run cycle	C+
Hare idle	C+
Hare jumping out of hole	C