

Game Design Document

Version 1.0

Sam Liveson

December 16, 2015

Table of Contents

[Design History 4](#_Toc438057275)

[Section I: Game Overview 5](#_Toc438057276)

[Game Concept 5](#_Toc438057277)

[Feature Set 5](#_Toc438057278)

[Genre 5](#_Toc438057279)

[Target Audience 6](#_Toc438057280)

[Game Flow Summary 6](#_Toc438057281)

[Look and Feel 6](#_Toc438057282)

[Project Scope 6](#_Toc438057283)

[Gameplay and Mechanics 6](#_Toc438057284)

[Gameplay 6](#_Toc438057285)

[Game Progression: 7](#_Toc438057286)

[Mission/Challenge Structure: 7](#_Toc438057287)

[Objectives: 7](#_Toc438057288)

[Play Flow: 7](#_Toc438057289)

[Mechanics 7](#_Toc438057290)

[Physics 8](#_Toc438057291)

[Raindrop computations: 8](#_Toc438057292)

[Raindrop initialization: 8](#_Toc438057293)

[Raindrop collision: 8](#_Toc438057294)

[Items: 8](#_Toc438057295)

[Mob: 9](#_Toc438057296)

[Movement: 9](#_Toc438057297)

[General Movement 9](#_Toc438057298)

[Other Movement 9](#_Toc438057299)

[Actions 9](#_Toc438057300)

[Picking Up Objects 10](#_Toc438057301)

[Game Economy 10](#_Toc438057302)

[Screen Flow 10](#_Toc438057303)

[Replaying and Saving 10](#_Toc438057304)

[Story, Setting, and Character 10](#_Toc438057305)

[Back Story 10](#_Toc438057306)

[Game Progression 11](#_Toc438057307)

[Game World 11](#_Toc438057308)

[Characters 11](#_Toc438057309)

[Woman of Windsor 11](#_Toc438057310)

[Levels 11](#_Toc438057311)

[Salem Forest 11](#_Toc438057312)

[Synopsis 11](#_Toc438057313)

[Objectives 11](#_Toc438057314)

[Physical Description 11](#_Toc438057315)

[Critical Path 12](#_Toc438057316)

[Closing Material 12](#_Toc438057317)

[Salem Beach 12](#_Toc438057318)

[Synopsis 12](#_Toc438057319)

[Objectives 12](#_Toc438057320)

[Physical Description 12](#_Toc438057321)

[Critical Path 12](#_Toc438057322)

[Closing Material 13](#_Toc438057323)

[Interface 13](#_Toc438057324)

[Menus 13](#_Toc438057325)

[Starting Menu 13](#_Toc438057326)

[Live Console 13](#_Toc438057327)

[Modal Menu 13](#_Toc438057328)

[Lighting Models 13](#_Toc438057329)

[Artificial Intelligence 14](#_Toc438057330)

[Opponent AI 14](#_Toc438057331)

[Player and Collision Detection 14](#_Toc438057332)

[Technical 14](#_Toc438057333)

[Game Art 15](#_Toc438057334)

[Management 15](#_Toc438057335)

[Appendices 15](#_Toc438057336)

[Asset List 16](#_Toc438057337)

# Design History

* December 15, 2015:
	+ Add Modal Panel
	+ Add Desert Scene
	+ Add Start Scene
* December 14, 2015:
	+ Add to Panel: HealthText, DaysText, HighScoreText
* December 13, 2015:
	+ Add Side Canvas listing Wine, Quill, Clock, and Health
* December 12, 2015:
	+ Add Rain Drop Shadows and Collider
	+ Incorporate Wine, Quill, Clock, and Health objects into Forest Scene
* December 11, 2015;
	+ Detail Forest Terrain
	+ Add Rain Drop AI
* December 10, 2015:
	+ Add Cauldron to Forest Terrain
	+ Add Mob and Mob AI
* December 9, 2015:
	+ Experiment with Projector Shadows for Raindrop
* December 6, 2015:
	+ Add Blood Mage Jones
	+ Create Flat Terrain
	+ Add Raindrop

# Section I: Game Overview

## Game Concept

Witch’s Brew is a third person arcade-style game that involves strategy, humor, and character. The game is centered around the modern American’s obsession with mythical creatures, as demonstrated by the Harry Potter and Twighlight catalogues.

The setting is Salem Massachusettes at the height of the Salem Witch Trials. The witch, by the name of Alice Young (the Woman of Windsor), travels along the forests of Salem and its sandy beaches.

In Witch’s Brew, we control the Woman of Windsor as she battles a typical day shopping for ingredients for her cauldron. From ahead, she must avoid water as the rain beats down day after day, and from behind, she must avoid a nervous but steadfast mob that is slowly encroaching upon her. All the while, she must seek and collect the four ingredients in her most tasty potion: blood from meat, the ink and feather from a quill, the time from a clock, and the alcohol from wine. Once she has collected all necessary ingredients, she can make her way to the safety of her cauldron.

The arcade facet of Witch’s Brew comes from the fact that we are ultimately dodging objects and competing for the highest score. The witch who survives for the longest number of days shall have her name live on in infamy.

## Feature Set

* Woman of Windsor
* Obstaces
	+ Raindrops
	+ Mob
* Two Levels
	+ Salem Forest
	+ Salem Shores
* Potion Ingredients
	+ Meat
	+ Clock
	+ Quill
	+ Wine
* Goal
	+ Cauldron
* Data
	+ High Score
	+ Day Counter
	+ Health Counter
* Game Over Menu
	+ Restart
	+ Quit

## Genre

Witch’s Brew is an adventure-style role-playing game. The player takes on the role of the Woman of Windsor as she avoids a mob of angry townspeople trying to have her burned at the stake, and to avoid rainy days. The adventure is exploring the two levels included in the game and immersing oneself in the magical forest and beautiful beach. The arcade component comes from the requirement to dodge objects (rain) while collecting other objects. The technical skill required to avoid obstacles is more reminiscent of games such as Pac-Man, albeit on a 2-dimensional scale.

## Target Audience

The game is targeted for audiences 8 and up because of strong subject matter. The Salem Witch Trials involved mysterious forces, including the fear of supernatural beings, and gruesome tragedy in the literal killing of innocent civilians. While Witch’s Brew is not inherently violent, the further research the game inspires covers a subject matter not appropriate for children under 8 years of age.

## Game Flow Summary

The player starts with an open level ahead of them. The player is aware of the game objectives because the start page describes finding the cauldron at the other end of the level. Levels are four times as long as they are wide. Additionally, the player cannot turn around because the shadow of a mob covers the entire length of the level behind them, and contacting the mob collider ends the game. Hence, movement toward the cauldron is encouraged and necessary for the player. While moving forward, the player must move left and right around the level to contact items to use with the cauldron. Upon reaching the cauldron, the player is moved to a new level with a faster mob behind them. The game continues until the player runs out of health, either by melting or by contact with the mob, or if the player reaches the cauldron without all necessary items.

## Look and Feel

While the game is about the typically dark and mysterious topic of witchcraft, the game maintains a light atmosphere and a slight sense of urgency. The raindrops are occurring on an otherwise sunny day. The forest and beach levels are easily recognizable as common terrains that could occur anywhere. There are no scary unknowns involved in these levels. The game maintains a lighthearted feel with a sense of urgency that adds fun to the game.

## Project Scope

Number of locations: 2

Number of levels: 2

Number of NPCs (non-player characters): 0

Number of weapons: 0

Number of collectable items: 4

# Gameplay and Mechanics

## Gameplay

Game Progression: Upon reaching the cauldron with all required items, the player gains +1 health, meaning they can withstand one new raindrop collision, and the number of days the player has survived increases by 1. However, each new level gets slightly more complicated in that the mob behind the player moves forward faster and faster. In the first level, the mob moves at an almost indiscernible speed (when not keeping up with the witch). This speed increases more dramatically at each level so that the player has to keep this in mind. Additionally, the number of items available to the player decreases to two of each item (meat, clock, quill, and wine) after several levels. In the first level, the player has four of each item available.

As the player contacts raindrops, his or her health diminishes by -1 each time. When the player dies, either by melting or mob, or reaches the cauldron without all required items, the player can restart. The high score (number of days survived) is recorded by the game.

Mission/Challenge Structure: The challenge is finding all items (which fall randomly) while staying ahead of the mob creeping up behind the witch, and avoiding raindrops. Each level is designed to be traversed by the player in 30 seconds in a straight shot; however, the collection of items and movement to avoid raindrops increases the amount of time a level may take.

Objectives: The objective of the game is to stay alive for as many levels as possible by avoiding obstacles while collecting necessary items to move to the next level. The player has motivation to achieve the highest number of days surviving. Moments of elation come when items are collected successfully, and raindrops are avoided successfully, and when the cauldron is reached successfully.

Play Flow: Play flow is based on a built-in Unity Third Person Camera and Third Person Controller. The camera generally remains behind the character and moves smoothly with the character. The flow from one level to the next is instantaneous; there is no option to rest while playing. The player has no option to pause except to lose the game by backing into the mob behind them. This is to avoid unfair high scores based on players taking long breaks to refocus. However, upon losing the game, a useful pop-up menu appears with a unique explanation of the loss, and the option to restart or quit.

# Mechanics

Game physics are more intelligent than meets the eye. In order to allow the player to move anywhere forward or side-to-side on a large level, many raindrops are constantly initiated at random locations throughout the scene. The resulting scene of raindrops as far as the eye can see is awe-inspiring. As described below, computing effort is minimized by only instantiating raindrops in front of the player, and by only enabling colliders and raindrop shadows if the raindrop is within 5 meters of the player in all directions. Item colliders are generally spheres to save computing effort, and player movement scripts are taken from Unity.

## Physics

Raindrop computations: It is always raining in Salem. However, the rain doesn’t have to slow the game down (although I recommend playing in performance mode). Rain development is intelligently designed to save computational effort. Raindrops from 3DWarehouse were chosen because of their low number of polygons. These raindrops were compared in performance to basic 3D game object spheres in Unity. No discernible difference was noticed. However, on enabling colliders and projection shadows on all raindrops, performance decreased significantly in terms of frames per second. A solution was to selectively enable projectors and colliders only on raindrop objects within a 5 meter radius from the player in plane directions (and within 1 meter behind the player). Each update function continuously checks all raindrop locations. If the raindrop is located within 5 meters of the character’s location at that time, colliders and projectors are enabled. Colliders are not disabled at any point. However, raindrop objects are destroyed when they travel below the terrain.

Raindrop initialization: Raindrops are initialized randomly. Raindrop location is based upon the witch’s location in each terrain. Raindrops are not initialized behind the witch. Raindrops are initialized randomly for the entire width of each level, and for the length of the level from the witch’s location to the end of the terrain. In order to keep raindrops from becoming more dense as the length of the level decreases, raindrop initialization rate is based on the witch’s location. At the start of the level, 17 raindrops are initialized every time the DropDrop() function is called. As the witch’s location along the z-axis (the length of the level) reaches 1/17ths of the level, 2/17ths, etc., the number of raindrops initialized decreases by that amount. By the time the witch is near the cauldron, only one or two raindrops is being initialized every time DropDrop() is called. However, because they are only being initialized on a slim length of terrain, their density is similar to that of when the witch began the level.

Raindrop collision: Raindrops have spherical colliders with a similar radius to the raindrop mesh. The pointed top of the raindrop has no collider. Upon enabling collision, raindrops do not collide with terrain or with themselves. All collisions are programmed in the Witch.cs script attached to the Main Character. Upon the witch colliding with the raindrop, if the colliding object was tagged a “drop”, the colliding object is destroyed, and the health of the witch decreases by 1.

## Items:

Items are instantiated upon the level startup. They appear randomly within the range of the level width and height in front of the witch. The objects have rigidbodies with gravity enabled; however, they only are able to move in the y-direction (height). So upon initialization, the object falls to the height of the terrain and does not move in the x- and z-directions from there.

Upon the witch colliding with an item, the witch checks its tag in a series of “if” statements. The corresponding item is destroyed, and the corresponding text box in the canvas is updated. If multiple of the same item is collected, the canvas reprints the same message (the item name in all-caps, followed by a bracket indicating that item is query is full.

Subsequent levels contain fewer items to increase difficulty. The original level contains four of each item; several subsequent levels contain three of each item; finally, the hardest difficulty achieved after several subsequent levels is an initialization of two of each item in the scene. This is more difficult because once an item is passed, there are not subsequent items to collect, and the game is essentially over. Items are designed to be large enough to be seen from anywhere on the level, but trees and rocks do inhibit their view.

## Mob:

In order to improve game performance, the mob was decided not to be composed of actual character AI. Instead, the mob is a shadow projection of a mob that includes an invisible plane collider at the back of the projection. The collider and projector move together. Movement is calculated according to the following algorithm called in the mob’s Update() function on each frame: if the witch and mob are farther than a designated distance apart, the mob moves forward to the beginning designated distance. This accounts for the quick movement of the mob to stay a designated length behind the witch as she moves forward. If the mob and witch are closer together, the mob slowly moves in the z direction (towards the witch). The speed at which the mob moves increases in each new day. This mimics the movement of a terrified mob of people; they will run up to the witch, but once they get close, they frighten and only move slowly towards her.

Although the mob does not move in relation to the camera, the camera following distance from the witch is such that the mob shadow is generally discernible, but the end of the mob is never seen through the camera; this provides an element of unknown and imagination. The player imagines that the projection is formed by a shadow of people objects.

## Movement:

### General Movement

Witch movement is controlled by built-in Unity scripts for third-person control. General control is completed using the directional keypad and the spacebar to jump. The level is designed so that the player cannot jump higher than the cliffs surrounding the forest, or the colliders surrounding the beach.

### Other Movement

Rain movement is defined by Rigidbody Gravity. It was decided that the gravity component will remain constant while incrementing through each new level to avoid player frustration at attempting to predict how to dodge the raindrops.

## Actions

As described above, witch movement is described by the built-in Unity third person controller script using the directional keypad and spacebar to jump. Version 1 of the game does not include the option to fly or increase witch speed.

### Picking Up Objects

Picking up objects is accomplished simply by colliding with the object. The object is destroyed. The canvas on the main screen indicates that an item is picked up by capitalizing that item in the canvas and indicating that its requirement is [FULL]. Items still required are listed in lowercase letters and designated [empty].

### Game Economy

The player must decide their strategy to survive another day in Witch’s Brew. Initially, mob movement is slow enough that patience is crucial to dodging raindrops and deciding a course of action before moving foward. Eventually, mob movement forward occurs more quickly so that instinctive and riskier movement throughout the level is important to balance with the collection of items. The game hinges on a balance between forward movement and careful item collection. There may come a point where collision with a raindrop is necessary to avoid being taken by the mob from behind.

## Screen Flow

Start Scene: Initial scene. Includes player instructions and a “Start” button.

Forest Scene: Player travels through a forest level. This is the first level in the game.

Beach Scene: Second level. Player travels through a sandy beach level surrounded by water.

Modal Menu: Full-Screen Modal Canvas. Appears upon the game ending. The text of the menu differs depending on the event that occurred. Player has the option to restart or quit.

## Replaying and Saving

The one saved item in the game is the High Score component. This is saved in the Player Settings. The game can be replayed upon losing all health by pressing “Yes” when prompted to replay.

# Story, Setting, and Character

## Back Story

Witch’s Brew has an animated story as its history. It is essentially based on true aspects of history. In 1963, the Salem Witch Trials of Salem, Massachusetts were underway. One of the woman accused was Ms. Alice Young, the Woman of Windsor. Salem had woods (Salem Woods), and it has a port area. The game takes place in two levels that represent these areas. As a witch, Ms. Windsor was hung by the city for witchcraft. In Witch’s Brew, she is chased by a mob. Where Witch’s Brew

## Game Progression

The game gets harder as it progresses. As the player moves through levels, the mob moves more quickly, more raindrops fall, and less items are available to collect. Upon restarting the game, these changes to settings are reset to their original (easier) values.

## Game World

As described above, the game world consists of a rocky beach and a wooded forest. All levels are connected via the cauldron; its ability to cast potions is also magical in that it transfers the witch from one world to another back and forth.

## Characters

Woman of Windsor is able to walk, run, jump, crouch, and perform other animations courtesy of Unity Standard third person character assets. Her back story is described in previous pages; she is trying to survive the daily grind by collecting her potions each day and making it to her cauldron. At that point, another day is done, and she moves on to the next potion hunt and balancing act while dancing in the rain. Unfortunately, it is always the norm in the day of the life of a witch.

# Levels

The two levels created thus far for Witch’s Brew are Salem Forest and Salem Beach.

## Salem Forest

### Synopsis

Salem Forest is the starting level of our game. The terrain is long and narrow; surrounding us are cliffs that act as colliders so that we don’t run off the level. Varying types of trees surround the area.

### Objectives

Unique aspects of this level is that trees may come between the character and camera so that it is difficult to see where droplets are at any given time. Trees can be long and wide so that it is difficult to see where you are traveling. Additionally, the dark grass floor makes it difficult to see raindrop shadows so that you are more likely to collide with them.

### Physical Description

The level was designed with the Terrain Toolkit C# script to have rolling hills. Varying trees were selected and mass placed in a density to achieve our final forest balance between shrubbery and open space. Edges of the level were raised manually to act as colliders between the character and life beyond the level.

### Critical Path

The character is pushed straightforward. This design was chosen to avoid any confusion in what path the character should travel. The character is physically unable to move any direction except forward; a mob collider will activate if she backs up, and the level walls prevent movement left and right. The player continues to move forward and the mob continues to cut off the level behind the character.

### Closing Material

The cauldron is not visible through the trees until the player reaches the other end of the level. The cauldron is slightly grown into the terrain to add a natural and worn look to the level. The player has been told in the instructions to collide with the cauldron and complete the level.

## Salem Beach

### Synopsis

Salem Beach is the second level of our game. The terrain is long and narrow; surrounding us are invisible plane colliders so that we don’t run off the level into the water. Rocks with built-in colliders are spread throughout the area. Water continues in all directions as far as we can see.

### Objectives

Unique aspects of this level is that it is wide open and very easy to see the entire level. It is a generally considered “easier” than the forest level in that aspect. A player can train on this level and generally end up with additional health to prepare to go back into the woods.. Trees can be long and wide so that it is difficult to see where you are traveling. The sandy terrain floor makes it easy to see raindrop shadows so that you are more likely to avoid them. However, the sand is difficult to move through; movement is slower than in the grassy forest.

### Physical Description

The level was designed with the Terrain Toolkit C# script to have rolling sandy beaches. A rock mesh from the Unity Standard Assets was selected, a spherical collider was added to create the final rock prefabrication, and the prefabrication was mass placed in a density to achieve our final beach balance between rocky areas and open space. Four panel colliders were placed manually at the land/water interface to prevent the character from getting stuck in the water; witches cannot swim.

### Critical Path

The character is pushed straightforward. This design was chosen to avoid any confusion in what path the character should travel. The character is physically unable to move any direction except forward; a mob collider will activate if she backs up, and the level colliders prevent movement left and right. The player continues to move forward and the mob continues to cut off the level behind the character.

### Closing Material

The cauldron at the end of the level is slightly buried in the sand to add a natural look to the level. The player has been told in the instructions to collide with the cauldron and complete the level.

# Interface

## Menus

### Starting Menu

The starting menu is created from a panel that spans the entire scene. The background sprite added to the panel provides instructions on how to play the game. One button to start the game is present on the panel; upon clicking, the player is brought into our first forest level.

### Live Console

A simple but critical console is present while playing the game, constantly available for the player’s reference. This console displays what items the player still has to collect, how much health the player has, for how many days they have survived, and the high score for how many days any player has survived. Part of the game is to be able to balance looking at the score panel to see what items are needed while still avoiding raindrops and staying ahead of the mob. You don’t want to watch the health tick down as you’re pounded with raindrops!

### Modal Menu

A modal menu appears in three instances:

1. When a player has lost all health. ("You have lost all health!\n\nDo you want to play again?")
2. When a player has collided with the mob. ("The mob got to you!\n\nDo you want to play again?")
3. When you have made it to the cauldron but you don’t have all items. ("You did not collect all items! Your potion cannot be completed.\nPlay again?")

In each instance, the player has the option to restart the game by pressing the “Yes” button, or to exit by pressing the “No” button.

### Lighting Models

Rather than adding light to the game, Witch’s Brew is constantly adding shadows. The mob is a shadow projector that follows the witch as she moves. Each raindrop has a shadow projector that is enabled as the witch approaches the raindrops.

# Artificial Intelligence

## Opponent AI

While there are not active opponents, there are passive obstacles that behave intelligently. Raindrops only fall in the vicinity of the character, and more fall as the player advances to each next level. The mob behind the character keeps up with her and advances more quickly as the player progresses through levels. Items are spread throughout the level randomly, and less items are availbable in each subsequent level. Each of these objects have associated colliders and actions that must take place upon collision; whether it’s updating the Live Console to lose health or report that another item has been collected, or to indicate that the game is over and pass the correct Modal Menu to the character.

## Player and Collision Detection

Collision detection is an integral component of every part of Witch’s Brew, as listed below:

1. Collisions determine that the witch has hit a raindrop
2. Collisions allow the player to stay on the terrain
3. Collisions are how the player collects items
4. Collisions are how the player avoids walking past the main level path
5. Collisions are how the player ends each level
6. Collisions are how the player avoids trees and rocks

Each collision method has an associated “if” statement present in Witch.cs to perform the required functions, including loading the next level.

# Technical

The game was developed on a Windows 8 laptop using Unity Personal 64 Bit. The scripting language was C# in Visual Studios 2013. The Start Screen Sprite was developed in Microsoft Power Point. All 3D game objects besides the cauldron were imported from 3D WareHouse or Unity Asset Store as free items.

The game is designed to be capable of being played on even a Windows x86 laptop. The player only needs a keypad and mouse to play. The game is recommended to be played in a format that emphasises performance over artwork because of the memory and computation required with constantly displaying more raindrops.

Character movement was provided by scripts from the Unity Standard Assets Third Person Character and Third Person User Control Scripts. Camera scripting was performed by the Free Look Cam and Protect Assets From Wall scripts from the Unity Standard Assets. Other scripts were developed using basic kinematic movement.

# Game Art

Besides the cauldron, game art was incorporated by three places:

* Unity Standard Assets
* 3D Warehouse
* Unity Online Store

Please see the Art Bible in the Appendix for more information.

# Management

Working individually was both a blessing and a curse. The blessing was enhanced and quick decision making. The curse was less ideas and creativity in the development process, as well less programming expertise. Game Development was balanced with full-time work and conflicting course requirements.

Game Development Landmarks:

* December 15, 2015:
	+ Add Modal Panel
	+ Add Desert Scene
	+ Add Start Scene
* December 14, 2015:
	+ Add to Panel: HealthText, DaysText, HighScoreText
* December 13, 2015:
	+ Add Side Canvas listing Wine, Quill, Clock, and Health
* December 12, 2015:
	+ Add Rain Drop Shadows and Collider
	+ Incorporate Wine, Quill, Clock, and Health objects into Forest Scene
* December 11, 2015;
	+ Detail Forest Terrain
	+ Add Rain Drop AI
* December 10, 2015:
	+ Add Cauldron to Forest Terrain
	+ Add Mob and Mob AI
* December 9, 2015:
	+ Experiment with Projector Shadows for Raindrop
* Decembr 6, 2015:
	+ Add Blood Mage Jones
	+ Create Flat Terrain
	+ Add Raindrop

# Appendices

## Asset List

* Witch
	+ BloodMageJonas humanoid
	+ Unity 3D Asset Store
	+ <https://www.assetstore.unity3d.com/en/#!/content/7004>
	+ Scripts:
		- Unity Third Person User Control
		- Unity Third Person Character
* Cauldron
	+ Custom-made 3D GameObject
	+ Constructed of Cylinder, Sphere, and fire material
* Terrain
	+ Designed using Terrain Toolkit
* Rocks
	+ From Unity Basic Assets
	+ Sphere collider added
* Trees
	+ From Unity Basic Assets
* Mob Shadow
	+ 3D Warehouse 2D component “Mob Silhouette”
	+ <https://3dwarehouse.sketchup.com/model.html?id=1d438d2bb1987bea2d50efd1d53ca470>
	+ Input as Lightmap and used as a Projector
* Rain Drop
	+ 3D Warehouse component
	+ <https://3dwarehouse.sketchup.com/model.html?id=6d90a47d365e26926f818f254c82cbd4>
* Meat
	+ Unity Asset store Meat Pack
	+ <https://www.assetstore.unity3d.com/en/#!/content/19234>
* Wine Bottle
	+ 3D Warehouse Red Wine Bottle
	+ <https://3dwarehouse.sketchup.com/model.html?id=a452a6dbc209dc0422833d1a51642f17>
* Clock
	+ 3D Warehouse Wall Clock – Wooden Round
	+ <https://3dwarehouse.sketchup.com/model.html?id=fb0dbe220131e28f6402b8f491cd92c7>
* Quill
	+ 3D Warehouse Quill and Dipping Bottle
	+ https://3dwarehouse.sketchup.com/model.html?id=a0dc89f1f77029064e6bf0ce5b328f41