Viscosity η

Developers:

Aidan Lawrence

Julio Gonzalez

Phillip Gonzales

Zac Bogner

AMERICAN MATERIAL RESEARCH LABORATORY

EXPERIMENT SUBMISSION DATE: MARCH 21 1955

Technical Design Document

APPROVAL PENDING

TDD Version 1.0 --- MAY 17th 2015

Table of Contents

Project Overview

Game Concept

Viscosity is a first person physics puzzler that allows the player to immediately manipulate the state of a strange sample of scientific matter known as "Viscosium." Players will take the role of a scientist testing the marketability of this new matter sample, performing tests and puzzles utilizing the properties of two Newtonian matter states. The difference between this in-game sample of matter and a real-life non-Newtonian fluid is the ability to instantly change between states indefinitely, instead of a temporary state change.

Technical Goals

- Create an interactive "fluid blob" using the Unity interactive cloth renderer that players can easily move around and solve puzzles with.
- Seamlessly allow players to transition between solid and liquid Viscosium.
- Create a fully-fledged first person character controller capable of reacting to advanced terrain situations.
- Develop a system to control the Unity interactive cloth component.

System Requirements

***Viscosity* is a PC-only title**

Operating Systems Supported:

- **Windows**: XP, 7, 8, 8.1, 10
- **Mac OSX**: 10.7+
- **Linux**: Ubuntu 10.10 & SteamOS

Minimum System Requirements:

- Quad-core CPU running at 2.4 GHz or higher.
- Nvidia GTX 660 with 1GB of VRAM or higher.
- 4GB RAM.
- 1GB of free hard drive space.

Recommended System Requirements:

- Quad-core CPU running at 3.4 GHz or higher.
- Nvidia GTX 960 with 2GB of VRAM or higher.
- 8GB RAM.
- 1GB of free hard drive space.

Third Party Tools

Unity 4.6 Professional (Game engine):

Unity 4.6 is an “all-in-one”, multiplatform game engine with built-in support for interactive cloth rendering. At the time of *Viscosity’s* development, Unity 5.0 is available, however, it does not include the interactive cloth renderer critical to the core mechanic of this game.

http://unity3d.com/

Visual Studio 2013 Professional:

While Unity 4.6 is packaged with its own IDE, *Monodevelop, Visual Studio 2013 Professional* edition provides a more fully-fledged development experience.

https://www.visualstudio.com/

Adobe Photoshop:

Adobe Photoshop is an incredibly powerful, industry-standard image editing and visual effects processor. Photoshop’s huge feature set and versatility makes it the perfect image editing software package for Viscosity.

http://www.adobe.com/products/photoshop.html

Audacity:

Audacity is a free, open-sourced audio editor. Audacity provides the tools necessary to perform on-the-fly audio touch-ups, create samples and recordings, and even apply effects.

http://sourceforge.net/projects/audacity/

Autodesk Maya 2015

Autodesk Maya is a massive industry-standard 3D software package. Maya will be used to create 3D models for *Viscosity* as well as the UV maps used to texture said models.

http://www.autodesk.com/products/maya/overview

Tasharen NGUI

NGUI is a user-interface plug-in for the Unity game engine. All 2D interfaces in *Viscosity* will be rendered through NGUI.

http://www.tasharen.com/?page_id=140

Plastic SCM

Plastic SCM is a version control solution made to handle large binary files and not *just* scripts. It's fully featured code review system and excellent three-way merging technology makes Plastic SCM the ideal Unity development version control solution.

https://www.plasticscm.com/home.html

Gameplay

Major Player Actions

Players may:

- Move their character in any direction through the game world using the W A S D keys (or through any other axis-based input).
- Jump by pressing the SPACE key.
- Orient their character and move the camera by using their mouse.
- Change the state of Viscosium by getting within range of a sample, looking at it, and pressing their "Interaction" key (Defaulted to MOUSE LEFT and E).
- Control the movement of Liquid Viscosium by getting within range, looking at it, and holding down their "Influence" key (Defaulted to MOUSE RIGHT).
- Interact with specific world objects by pressing their "Interaction" key.
- Use all of these functions mentioned above to solve physics-based puzzles.

Physics

Excluding the unique physical attributes of Viscosium, *Viscosity* will feature standard Newtonian physics in 3D Euclidian space.

Player Controller Physics

Collision:

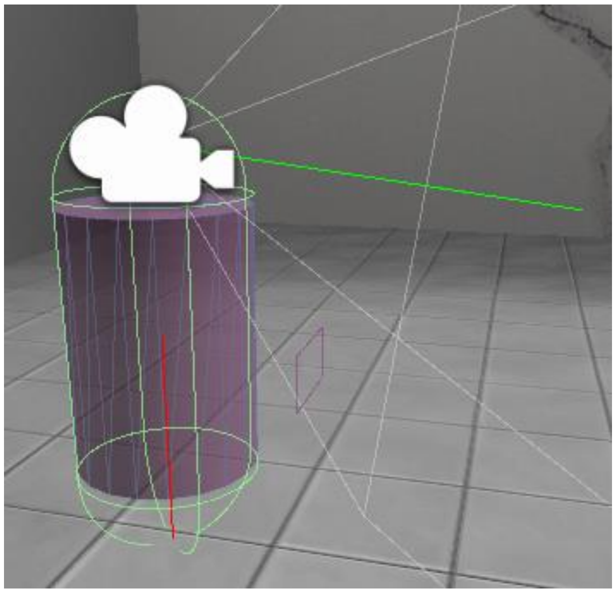
- The player's character controller uses a capsule-shaped collider that roughly fits the tangent of a 6-foot human being.
- The character controller will not be able to pass through any collider set to block objects on the "Player" layer.
- To prevent rigidbody friction from sticking the character controller to vertical surfaces, a special "non-stick" physics material will be added to any object with faces that exceed 80°.

Acceleration and Movement:

- The character controller will hook into a rigidbody solver and accelerate the player based on a linear interpolation between speeds over time. This rigidbody acceleration system prevents the character controller from clipping through walls or behaving strangely when colliding with other objects.
- The character controller will move at approximately 2 in-engine units per second while walking and can accelerate up to about 8 units per second when sprinting.
- The character controller can also jump on-demand by sending a vertical force command to the rigidbody solver to propel the player upwards.
- Players can rotate their character controller and camera by moving their mouse. Mouse movement on the X axis will rotate the character controller on the Y axis while mouse movement on the Y axis will rotate the camera on the X axis.

Sensory Features and World Reactivity:

- The character controller has a small, downward pointing raycast probe that has the ability to detect colliders below the controller and poll to see if the player is "grounded" or not.
- The character controller ground probe will also monitor the slope of any collider (normal) below itself and will react by adding a proportional force downward to provide a "slipping" effect of climbing up a steep hill.
- Protruding from the player's main camera is another raycast that is used to identify objects in the game-world and acts as a tool to interact with items in *Viscosity*.


An image of the character controller in *Viscosity.*

World Physics

- Gravity in *Viscosity* will be set to the standard 9.8 m/s^2.
- Friction will be handled by physics materials and will change depending on the specific needs of each surface.
- The Unity physics collision matrix will be used to prevent certain physics layers from interacting with one another.

Viscosium Physics

Solid:

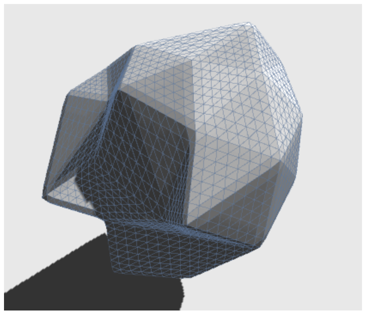
Solid Viscosium is simply a heavy, rigidbody controlled, environment sensing cube. Similar to the character controller, Solid Viscosium also has a raycast ground probe that will react to surfaces below it in certain situations. Should the cube detect that it is being moved forcefully and is touching the ground, it will emit scraping particles that mimic the color of the surface that it is on. Should the cube be dropped from a high place, it will emit a big “poof!” of dust particles. The rigidbody system is also set to resist rotational movement on the Y axis. All of these effects are to give the illusion of weight to the player, as Solid Viscosium is best known for being very heavy, dense, and difficult to move due to its high friction. Solid Viscosium will also react to fire by changing its color from a deep green to a bright red-white color, as if it is “heating-up”. Solid Viscosium also monitors for forceful impacts and will destroy itself catastrophically if hit too hard.

Liquid:

Liquid Viscosium is a wildly complicated fluid simulation utilizing the Interactive Cloth renderer.

The Interactive Cloth System:

In Unity 4.6, developers have the option of transforming meshes into an “Interactive cloth.” Interactive cloths are an odd extension of the PhysX physics engine library that will accept any mesh and average out the vertex locations while reacting to colliders in world-space. The result is usually best suited for items such as flags, rags, or any other object that fits the basic description of a cloth – any planar object that has relatively little compressive strength. However, in *Viscosity,* we have opted to use a very uniquely shaped, high-poly non-planar mesh that will serve as the base of our fluid simulation.



An in-engine look at the oddly-shaped mesh used for Liquid Viscosium.

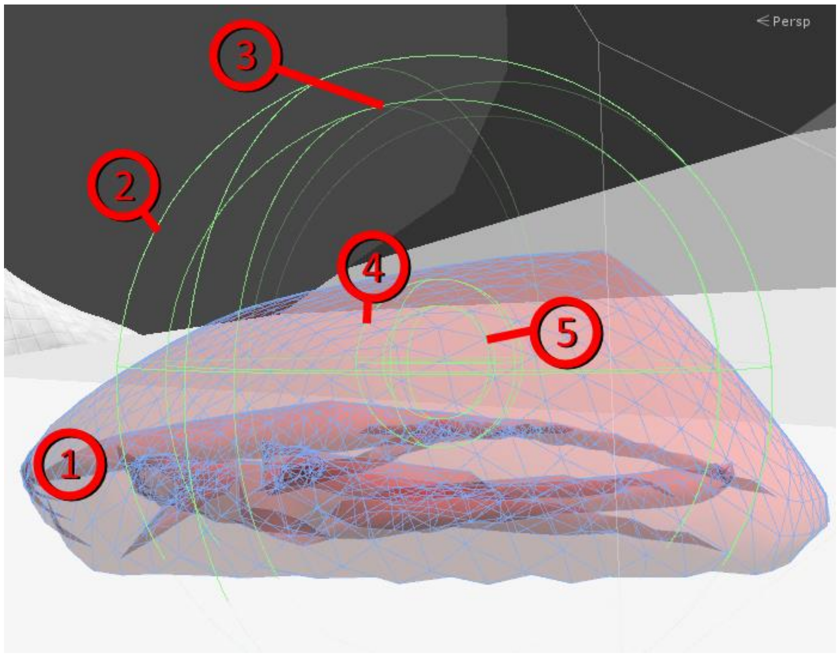
Settings to Achieve a Convincing Liquid Simulation:

Liquid Viscosium has a high stretching stiffness with a low bending stiffness to emulate a “rubbery-goopy” blob. The liquid is undampened with low friction settings to keep movement fluid and only semi-resistive. Internal pressure is set to be half of the outside atmospheric pressure in the scene (~14.7 PSI) along with a somewhat high density to keep the liquid viscous but still fluid like. Since Liquid Viscosium is a major interactive set piece, collision response is set to its default value of 1 due to strange behavior with other rigidbodies when modified. Tear factor is set at 0% chance (until it is time to destroy the liquid, of course!).

Collision:

Interactive cloth collision is very difficult to deal with as the cloth collision calculations are processed on a separate level of the PhysX library compared to the rest of the game world. This means that Interactive cloths *cannot* be detected by world triggers or collision checks natively. Therefore, a custom collision system needs to be hooked into the interactive cloth and perform the collision detection remotely.

Liquid Viscosium achieves true collision detection through an advanced matrix of external colliders that measure the location of the cloth simulation through the central bounding position of its renderer.



A look at Liquid Viscosium's collision in-engine

1) A false-colored version of our **liquid mesh** in mid-simulation. This mesh handles standard world collision and will react to rigidbodies without any modification. It cannot be detected by triggers or other colliders.
2) **The outermost main collision trigger.** This approximates its size by measuring the average radius of the farthest points in the mesh. It is used to detect outside triggers and to make the presence of the liquid known to any system looking to detect it.
3) **The particle collision trigger.** This collider is used to detect incoming particles that have the ability to affect the state of Liquid Viscosium (Fire, for example).
4) **The innermost main collision trigger.** This trigger is used for more precise Liquid Viscosium position detection. For situations that require the Liquid Viscosium to be completely colliding (liquid switches), the inner trigger provides a more accurate collision point relative to the center of the liquid.
5) **The acceleration bearing collider.** This collider is used to accelerate the Liquid Viscosium towards the point of player interaction. When not in use, the acceleration bearing will return to the center of the mesh.

Movement:

Yet another hurdle of interactive cloth simulations is their lack of a true transform system. Interactive cloths will begin simulating *at* their transform position, but will cease to be influenced by any new transform data once the simulation has started. The only method of directly moving an interactive cloth is through external acceleration.

When a player interacts with Liquid Viscosium…

- The player's camera-based raycast will report to the liquid that it is interacting with its external collider and pass along a Boolean value that represents the desired interaction behavior (grab or let-go).
- The Liquid Viscosium will then notify its control bearing collider to translate to the passed player raycast position by taking the cameras' forward vector, multiplied by 3, and added to the cameras' position.
- Once the bearing has been attached to the player's raycast, the liquid will then calculate the trajectory it needs to take by subtracting bearings' position from the liquids' cloth renderer bounded center position.
- At the same time, the liquid is also measuring the acceleration rate based on the distance delta between its rendering center and the collision bearings position.
- The Interactive cloth finally receives a final formulated acceleration vector and sets its "external acceleration" field to the vector value.
- To prevent exponential acceleration gain, the player will eventually "lose their grip" on the liquid if vertical acceleration is too high.
- Once the player has finished interacting with the liquid, or they have become out-of-range, the liquid will reset its external acceleration back to zero.

Special World Interactions:

There are a few special interactions with the world that Liquid Viscosium will respond to.

Fire Particles:

If the particle detection trigger encounters any particles tagged as “Fire”, the Liquid Viscosium clothe renderer will begin to swell up due to a substantial increase in internal pressure. Once the Liquid Viscosium reaches its pressure threshold, it will quickly pause its simulation, change its tear-factor to 100%, and resume its simulation immediately. The result is one big, goopy, Viscosium detonation.

Vacuum Tubes:

If the Liquid Viscosium outer collider detects that it is within interaction distance of a vacuum tube intake, it will begin to calculate an acceleration trajectory towards the center of the tube. If the Liquid Viscosium detects that its renderer center point is close enough to the vacuum center, it will temporarily destroy itself, invoke a delayed respawn method, and reappear at the vacuum tube outlet after the delay has passed.

Phase-Independent Viscosium Physics

This section will outline behavior that is found in both the solid and liquid versions of Viscosium.

Changing State:

To change the state of any Viscosium sample, the player must be both in-range and looking at the sample they wish to switch. As soon as the player hits their interaction key, the main player script will store the position of the current sample, destroy it, and respawn the opposite sample in its place. During this operation, there is a large puff of lightly-green-tinted particles and a couple brief animations to masque the swapping procedure.

Interacting with Energy Surfaces:

In a couple levels of *Viscosity*, players may notice specially colored “energy surfaces” that have interesting effects on their Viscosium sample. Should Liquid Viscosium touch the blue, low-energy surface, it will immediately “freeze” into Solid Viscosium. The opposite effect will occur if Solid Viscosium collides with the orange, high-energy surface – it will “melt” into Liquid Viscosium.

Other Physics-based Gameplay Elements

Viscosium Delivery Cranes:

These cranes are found in every stage and are used to deliver a single Viscosium sample (in solid form) to the player. Viscosium Delivery cranes can operate in any location and at any height (software locked to 200 meters high) and deliver a Viscosium cube on its mark perfectly, every time. Delivery cranes can be activated and deactivated (locked) remotely. Delivery cranes utilize a spring joint mechanism

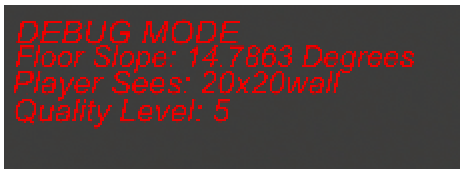
The Delivery Crane activation sequence is as follows…

- Check for lock. If not locked, continue…
- Ping ground from crane head and determine the distance.
- Subtract Solid Viscosium cube height from distance.
- Spawn a Solid Viscosium sample as a child object within the grippers.
- Deactivate the Viscosiums’ functionality until drop completes.
- Perform drop by interpolating the distance to the sprint joint “Max Distance” field.
- Draw the spring joint by creating a vector between the two attachment points and using the distance between the two points as a size scale on a line-renderer.
- Once drop the distance threshold has been met, detach the child Viscosium sample, reactivate its functionality, and trigger the “Open Crane Jaws” animation.
- Pause briefly, then retract back to the starting location and reset. Close doors.

Developer Tools

The Debug Menu

While in-game, developers (or curious players) may access the debug menu and all of its functionality by pressing the **HOME key five times**. When active, the debug menu will show up as a bright-red text field filled with information about the game. Accessing the debug menu also grants a multitude of in-game testing commands.



```
DEBUG MODE
Floor Slope: 14.7863 Degrees
Player Sees: 20x20wall
Quality Level: 5
```

Floor Slope: Refers to the angle of the normal (collider) that the character controller ground probe detects relative to the angle of the player. Basically, it detects what angle the floor is at.

Player Sees: If the player vision probe happens to be in-range of any object, it will read display the name of said object. If nothing is in range of the player's vision probe, this field will read "Nothing".

Quality Level: The current graphical preset.

Debug Menu Commands

Command	Activation Key
SPAWN LIQUID VISCOSIUM	L
SPAWN SOLID VISCOSIUM	O
DESTROY ALL VISCOSIUM	P
ACTIVATE SPAWN CRANE	C
INCREASE QUALITY LEVEL	=/+
DECREASE QUALITY LEVEL	-/_
LOAD STAGE 1	NUMPAD 1
LOAD STAGE 2	NUMPAD 2
LOAD STAGE 3	NUMPAD 3
LOAD STAGE 4	NUMPAD 4
LOAD STAGE TITLE	NUMPAD +
LOAD STAGE PROTOTYPE	NUMPAD 0

Performance

Scaling Based on Quality Levels

Because of the raw CPU horsepower required to render and simulate Liquid Viscosium, along with the rest of the scene, precautionary measures have been added to *Viscosity* to help ease the performance loss of using a cloth renderer.

Liquid Viscosium:

Liquid Viscosium will lower its poly-count by 10% (max 40%) for each quality level dropped. Quality level will also determine how often Liquid Viscosium updates its reflection.

World Quality:

Shadows, overall lighting quality, and texture quality will scale to player-determined quality settings.

Client quality settings can be found on the first splash-screen that opens before the program fully launches.

Rendering Techniques

Renderer

- Viscosity will be rendered using the Direct3D 11 pipeline.
- Only 16:9 displays will be officially supported. 16:10 Aspect ratio displays are only partially supported.

Texture Formats

- All textures in *Viscosity* should be of type .png, .jpg, or .tif.
- Shader choice is dependent on circumstance, however, some specialty shaders require extra inputs (normal maps, cubemaps, distortion maps, etc.).
- Ensure that all textures designed to be played on (the player or Viscosium will physically touch them) are set to advanced mode and that they are Read/Writable.

Specialty Techniques

Viscosity contains a few special rendering techniques.

Real-time Reflections

While there are several static reflection materials found within the levels of *Viscosity*, only the Liquid Viscosium material features a real-time reflection system. This is achieved by using a special camera placed within the center of the Liquid Viscosium that takes a cubemap "snapshot" of every direction and feeds it into the reflective shader on the cloth renderer. This cubemap snapshot is updated frequently, but not every frame – that would be far too costly. Instead, the cubemap snapshot update frequency scales up or down depending on player quality settings.

Anti-Aliasing

To cut-down on sharp texture "jaggies" found on geometry rendered on lower resolution screens, *Viscosity* passes its frame buffer through a FXAA2 filter. FXAA2 provides adequate anti-aliasing for nearly no performance cost.

Extra-Special Effects

To achieve the look of the opening cinematic, *Viscosity* used a variety of image effects. A sepia filter, noise filter, animated 2D overlays, and some clever camera swapping is how *Viscosity* managed to fit its entire opening sequence and title screen into one scene.

User Interface

NGUI

Viscosity uses NGUI as its primary user interface and 2D overlay tool. NGUI support forum can be found here: http://www.tasharen.com/forum/index.php?board=1.0
NGUI documentation: http://www.tasharen.com/forum/index.php?topic=6754.0

Title Screen UI

After the opening cinematic is rendered, *Viscosity* begins at the title screen user interface. Here, players can see the main game title, as well as three options.

- **New Game –** Starts a brand new game beginning from stage one.
- **Continue –** Continues the last available save game. If no save is available, this option is faded out.
- **Quit –** Completely exits the application.

In-Game UIs

Each scene that contains the character controller **must** include the following prefabs:

- **UI –** This is the main user interface object.
- **DEBUG_UI –** This is the user interface that pops-up when debug mode is enabled.

The Main In-Game User Interface:

The main user interface in *Viscosity* is designed to be simple, yet effective. In the center of the screen, there is a small square using a shader that will invert the color of anything behind it. This allows the player to have a permanently visible crosshair, regardless of background color. On the rare occasions where players need to be directly prompted with text, a minimalistic, translucent, black dialogue box with white text will brief the player.

The Pause Menu:

If the player hits the ESCAPE or RETURN key, in-game time will stop and the player will enter the pause menu.

In the pause menu, the player will see the following items:

- **Resume –** Exits the pause menu and resumes in-game time.
- **Restart Stage –** Restarts the entire stage from the beginning.
- **Main Menu –** Returns the player to the main title screen menu.
- **Quit Game –** Quits out of the entire application.

In addition to pressing the “Resume” button on the user interface, the player can also choose to hit the ESCAPE or RETURN key to resume the game.

Audio

Format

All audio, including sound effects and music tracks, **must** be encoded using the Ogg Vorbis format (.ogg).

Programming Style

- All scripts must contain a descriptive header comment. For example:

```
/*~~~~~~~~~~~TEAM NO FUN ALLOWED~~~~~~~~~~
 *~~~~~~~~~~~~~~~VISCOSITY~~~~~~~~~~~~~~~
 * Script: "LiquidViscosium.cs"
 * Script Programmer: Aidan Lawrence
 *
 * "This script handles 'Liquid Viscosium' and all of it's
 *  funky goodness."
 * */
```

- When using Plastic SCM, all changes should be thoroughly commented with clear, human-readable explanations of new or removed systems.
- Camel case will be used for all variables.
- Constants will be all capitalized with underscores for multiple-word identifiers.
- Functions, structs, enumerations, and classes will all begin with a capital letter.
- In-line comments should be used liberally to describe code-blocks, variables, and functions.

Example of proper variable and constant declarations:

```
int exampleInt = 0; //An example integer
const char CONSTANT_EXAMPLE = ‘c’; //An example constant char
```

Example of a proper function:

```
//This function serves as an example
void FunctionName(int overloadOne, char overloadTwo)
{
     //Logic goes here
}
```