Frank: The Digital Pet Game

CM0607: Games Case Project

Table of Contents

Project Management 4
Prologue: Roo 4
Team Management 4
Communication 4
Delegation 4
Scheduling 4
Tracking 5
Group Dissolution 5
Army of One 5
Motivation 5
Time Keeping 5
Publication 5
Review 6
Public Discussion 6
Work Environment 6
Tracking 6
Design 7
Summary 7
Player Actions 7
Props 8
Buildings 8
Frank 8
Mini-Games 9
Cooking 9
Cooking Version 1.0 9
Cooking Version 2.0 11
Plant 11
Tree 11
Fairy Catching 12
Fishing 12
Milling 12
Cooking 13
Interface 13
In-Game Interactions 14
Real World Interactions 15
XNA Game & Components 16
Initialize 16
Content Load / Unload 17
Update & Draw 17
Services & Components 17
Screen Manager 18
Debug Renderer 18
Problem: Render Order 18
Content Pipeline 19
Key Benefits 19
Models & Animation 19
Skinned Animation 19
Problem: XYZ vs XZY 20
Problem: Non-Identity Bind Pose 21
Problem: Multiple Animation Tracks 21
Custom Effect 21
Game Props 21

Problem: Separate Data 22
Renderer 22
Scene Graph 22
World 23
Problem: Special Case Objects 23
Problem: Multiple Render Passes 23
Base Object 24
World Attach 24
Model & Animation Object 24
Camera 24
Target Camera 24
Obit Camera 24
Problem: Polar & Spherical Coordinates 25
Sky Sphere 25
Shadows 25
Cloud Box 26
Water 26
Environment Mapping 27
Game Objects 27
Data Manager 27
Decoupling 28
Problem: Massive Switch 28
Game Prop 29
Building & Island 29
Button & Timer 29
AI 30
Networking 30
Progress Log 31
Player Actions 34
Props 35
Buildings 35
Mini-Games 35
Frank 36
Only one button! 0
Slow Progress 0
Still not happy with my progress 0
Hack our way through the Jungle 0
Island Scale Staying late at work to do university work is... 0
Progress and Elections 0
The Other Problem 42
You want HOW MANY! 42
Ye gods captain, PROGRESS. After the dismal run around the... 0
Day & Night Cycle Woot! Not only did I get a pretty awesome... 0
Shadows & Light 0
Lights Camera & Action 46
Bake It 46
Fake It 47
Flatten It 47
Deferred Volume 47
Shadows and Reworks So after a lot of thought and planning... 0
Shadows and Reworks 48
Almost 0
Distractions 0
Wooot! Progress, the shadow is low quality due to PC card... 0
To Debug it& Draw it Many people will tell you this but... 0
To Debug it& Draw it 49
Problem: Shadows giving strange LightViewSpace results 50
Shadows are EVIL! Okay above you see rough and blurred. The... 0

Shadows are EVIL! 52
Water 52
Cube Maps Ahoy 53

Management

Prologue: Roo

For the first semester and part of the second I was working on the Roo project with a team of three others.

Team Management

We opted to use Redmine & SVN as our Project Management software.

Communication

A key component of process management is keeping clear channels of communication. Two members of the group were employed full time while one other was employed part time so communication was vital.

We selected email as the primary form of communication. There was some discussion about using the Redmine forums as they are persistent and threaded but it was dismissed. A weekly meeting was also agreed upon to review.

Delegation

Due to the fractured nature of the group we needed to reduce dependency on each other's areas of work. For this reason we separated off technical elements of the project to each member.

Redmine Issue Catergories tracked this ensuring all new issues were correctly assigned.

Tasks were then broken down and assigned as issues on Redmine.

This was a key problem as for some technical areas I wanted to opt to work in tandem as choices about code layout would affect all systems, and require skeleton systems in place. Not being able to arrange this time to work as a group in one location hampered progress on core systems.

Scheduling

We laid out several goal markers, and marked these on our roadmap as deliverables. Each issue was assigned to a roadmap deliverable. The issues were also given rough dates which generated a gant chart.

Through the gant chart and roadmap a realtime view of the scheduling could be in place.

Tracking

Issue tracking, time logging and repository tracking were our key tools. As issues were updated and time taken logged for each update that would update all the tracking software enabling an overall view of the project.

All code and resources were stored in SVN repository to keep all members in sync. The commit comments and change log provide a key information point.

Group Dissolution

I am unable to comment on the circumstances or speculate on the dissolution of the group. I was simple informed it that I was no longer in the group due to personal reasons, with no further explanation.

I find this highly unprofessional as in a working environment personal or professional disagreements are handled as a two-way process by HR. Discussions and meetings are held, reasons given and an attempt to continue to work is made. Only as a last result is a employee reassigned to a different project or fired. At all times a legal reason must be given.

I have been on both sides of these situations, and as a witness to proceedings at several different companies, and in every case a resolution was found.

Army of One

While team management is no longer required for a one person project there are several other challenges.

Motivation

A key factor for a one person team is motivation. No progress occurs unless you do it. Meaning low points in activity or output provide a negative feedback curve which demoralises and reduces performance.

Time Keeping

One simple way to improve productivity is to set time keeping. Track time spent and assign working hours. I enforced this later in the project by working in the office when working on the project, enforcing that work ethic.

Publication

Another key source of motivation which a team has which a single person does not is showing off progress to the team. The show and tell feedback loop, which not only provides a boost to moral of the team as a whole but provides a crucial review step (see more below).

The solution in this case was to log all progress to a public internet blog and publish my progress. Pointing friends and co-workers at it to re-enforce progress. Another side affect of this, which you normally get in a team is a poke or query when published progress is low.

Review

A real problem in a one person project, possible the biggest problem is the lack of review. There is no other individual to quality check, discuss or review your project.

External people can only provide a loose review as a significant investment is required before an informed opinion can be applied to discussion points or progress can be reviewed.

Also in a production environment either Code Reviews or Extreme Programming is used to ensure code is sane and complete. This simple quality assurance of passing more than one set of eyeballs over some code leads to significantly stronger code.

Unit Testing can be used but is only efficient on an established system with known behaviour. The benefit is not seen until the code reaches a certain point of maturity.

Public Discussion

I've achieved some review by publicly discussing my project & issues in IRC, forums and on my blog. Very little quality can be gleaned on project specific issues as the investment is too high for most public discussion. General issues which are common across projects, such as render items, are much easier as most people have a point of reference.

Work Environment

Being fortunate enough to work in a professional studio I was able to leverage some of the knowledge. Discussions are more personal, and involved. The individual works with you on a daily basis and knows your common failings, areas of weakness and strength. They are also by default more educated and speciliased than general public.

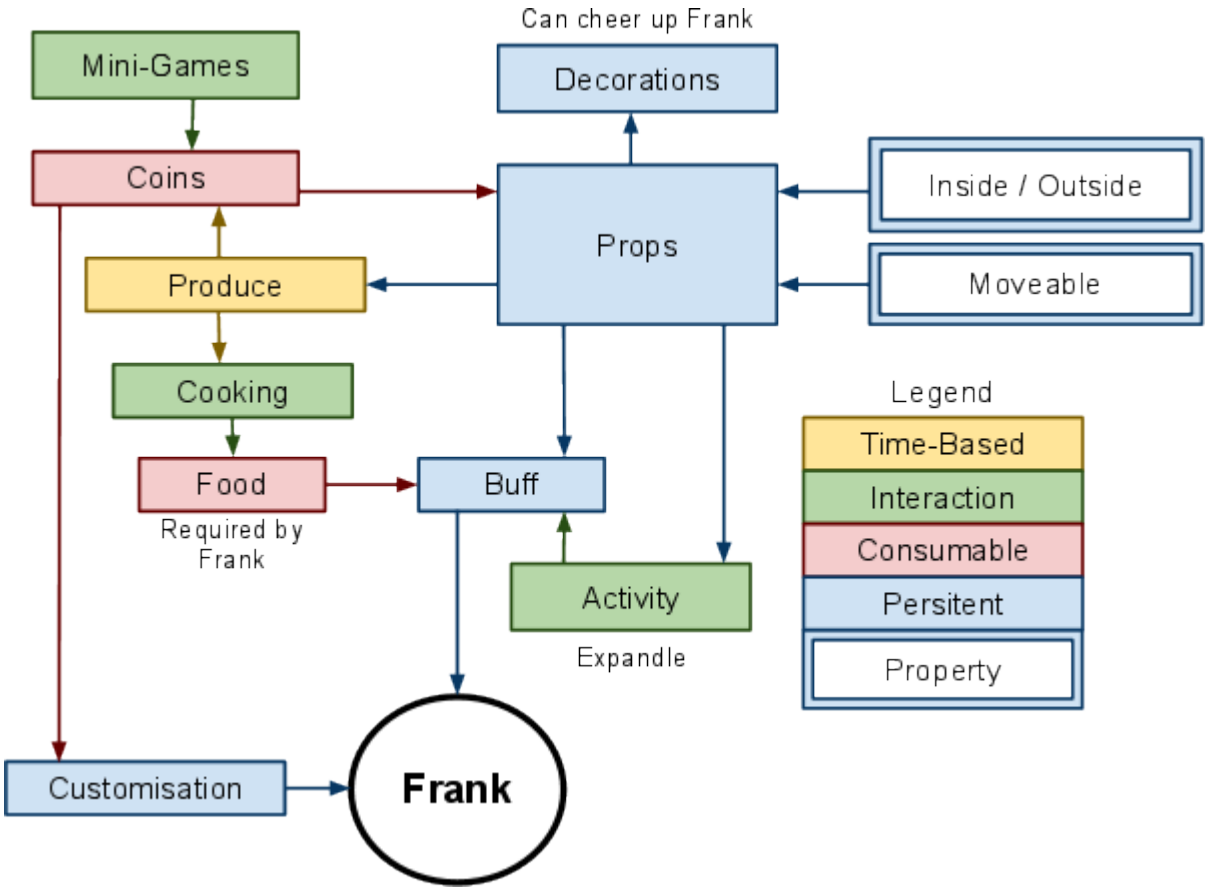
Tracking

Tracking of progress is done through SVN and the publication to the Tumblr Blog. See development diary. The main issue is the investment / reward of implementing a large scale project management software like Redmine is inefficient.

There lack of a need for team communication and delegation removes many of the benefits which would be gained from something like Redmine. Many features of the the project management can be replaced with the publication model.

Design

Summary



NO VIOLENCE!!!

Player Actions

- Frank
-
    - Pet / Scold
    - Feed Frank
    - Move Frank
    - Highlight Point
- Prop
-
    - Buy
    - Destroy
    - Move
    - Highlight

- Start Game

Props

- Bought with Coin
- Upgraded with Coin
- Properties
-
    - Inside / Outside
    - Moveable
- Production
-
    - One Off / Reusable
    - Input: ???
    - Output: ???
    - Time Based
    - Requires Activity
    - Max Capacity
    - Does not Spoil

Buildings

- Bigger on the inside
- Can be upgraded with Coin

Frank

- Carry
-
    - Only one item or tool at a time
    - Consumables dropped on the floor will disappear after time
- Stats
-
    - Strength
    -
        - Affects Milling
        - Affects Fishing
        - Affects Farming
    - Speed
    -
        - Affects Fairy Catching
        - Affects Fishing
    - Jump
    -
        - Affects Navigation
        - Affects Fairy Catching
    - Intelligence
    -
        - Affects Path Finding
        - Affects Learning Speed
    - Improving Stats
    -
        - Improved with Training
        - Can Degrade

            - Can be increased with Food

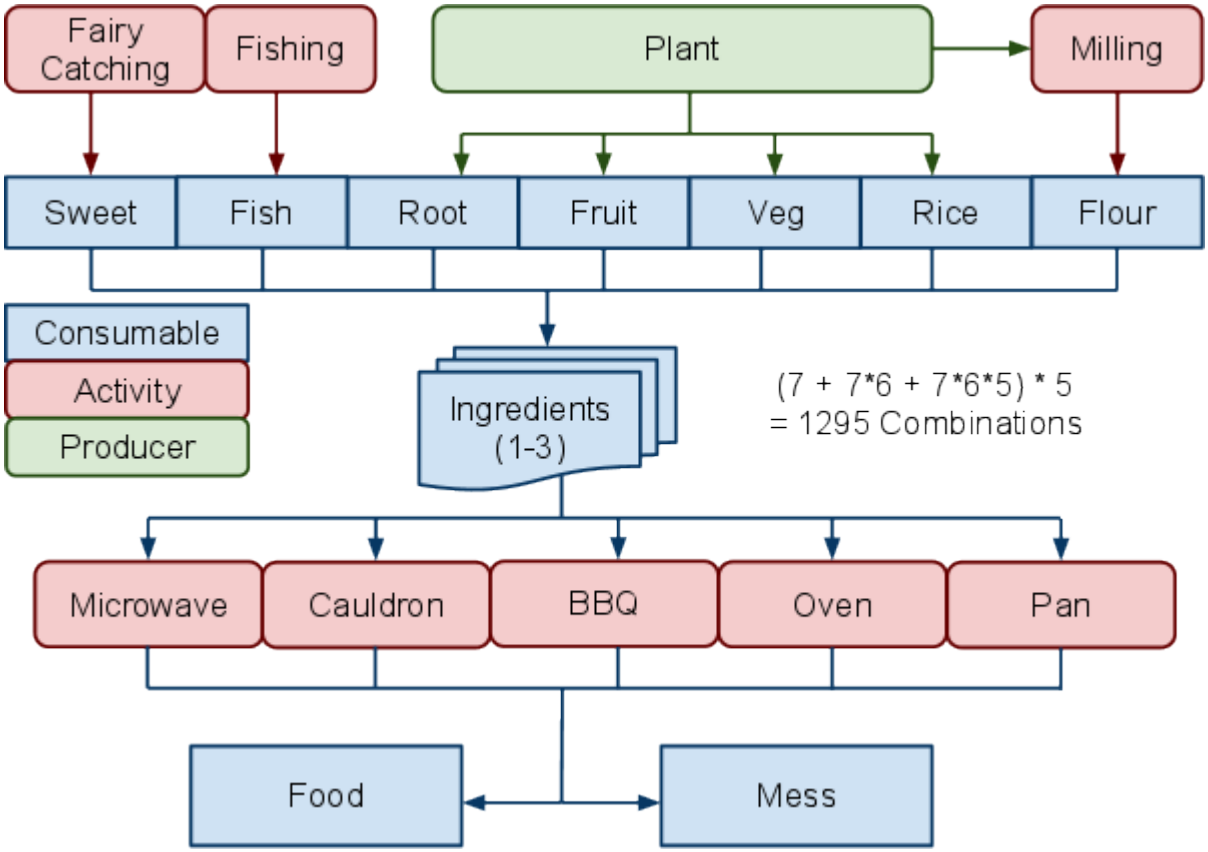
- Skills
-
    - Improved with Practice
    - Cooking
    -
        - BBQ
        - Cauldron
        - Oven
    - Milling
    - Farming
    - Fishing
    - Fairy Catching
- Knowledge
-
    - Recipes
    - .....
- Appearance
-
    - Pattern
    - Colour

Mini-Games

- Practice Mode
-
    - Teach Frank Skills
- Daily Tournament
-
    - Coin can be Won
- Race
-
    - Get from A-B in shortest time
    - Hurdles and Path Finding
- Treasure Hunt
-
    - Time Limit
    - Find X items in time limit
    - Find hidden items for bonus points
- Tag
-
    - Time Limit
    - Multiplayer only
    - One Pet is it
    - The others need to run or hide
    - On touch the target becomes it

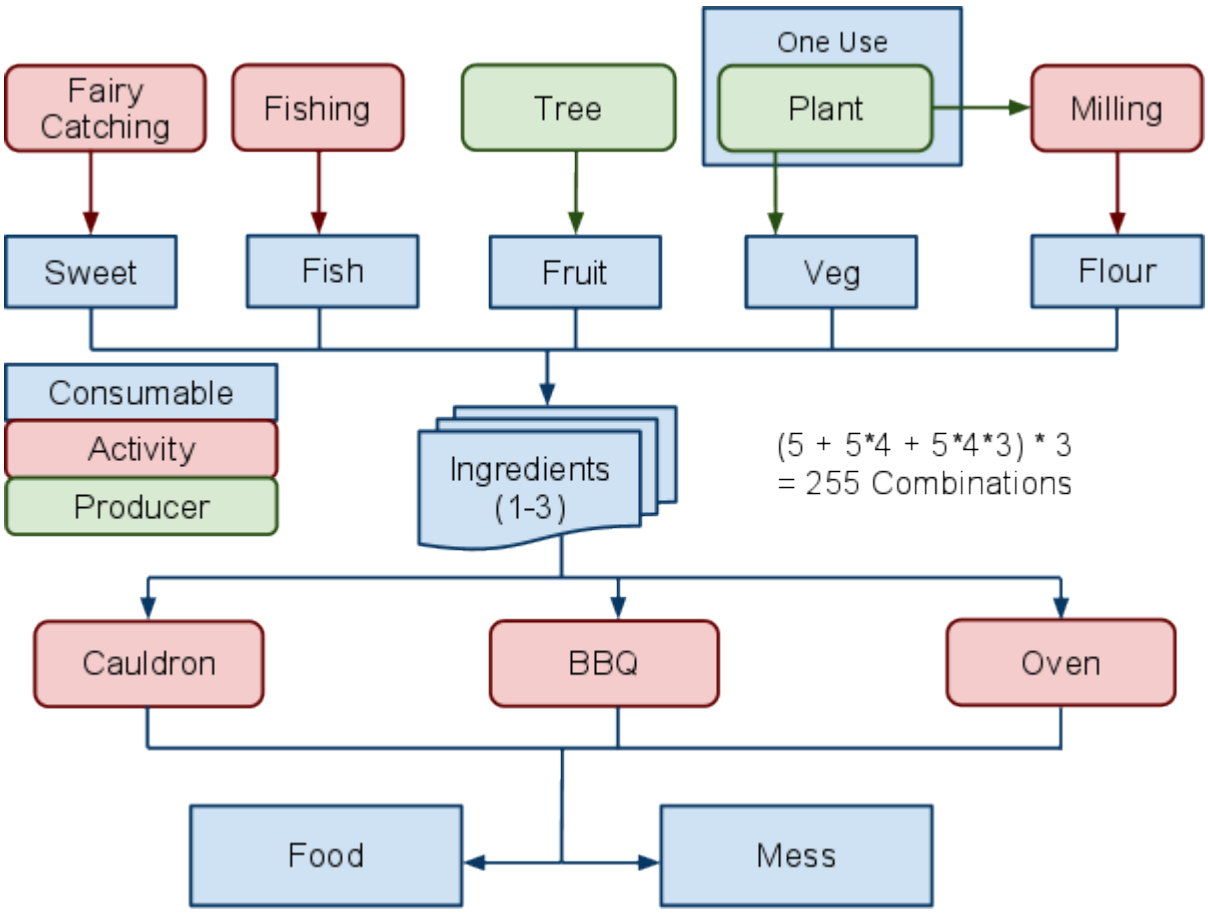
Cooking

Cooking Version 1.0



Dismissed due to large number of combinations!

Cooking Version 2.0



Plant

- Plow
-
    - Hoe Needed
- Plant
- Harvest
-
    - Destroys Plant
- Quality
-
    - Seed
    - Farming Skill
- Quantity
-
    - Farming Skill
- Colour
-
    - Seed

Tree

- Dig Hole

-
    - Spade Needed
- Plant
- Harvest
-
    - Remains for another Harvest
- Quality
-
    - Age
- Quantity
-
    - Farming Skill
- Colour
-
    - Seed

Fairy Catching

Can only be performed at night.
Catch a variety of different colour fairies.

- Quantity
    - Skill determines rate of Catch
    - Time of Night
- Quality
-
    - Skill affects quality
    - Butterfly Net
    - Time of Night
- Colour
-
    - Colour slowly change (i.e. rare for one type of colour to be out)
    - Red - Sunset
    - Blue - Midnight
    - Green - Sunrise

Fishing

Activity at pond.

- Quantity
-
    - Pond Slowly Re-stocks
    - Skill affects percentage of catches per bite
- Quality
-
    - Skill with Fishing
    - Fishing Rod
    - Bait
- Colour
-
    - Bait

Milling

- Any recipe that uses un-milled grain will come out burnt
- Quantity

-
    - Level of Mill
    - Skill with using Mill
- Quality
-
    - Quality of ingredients
    - Skill with using Mill

Cooking

- Recipe
-
    - Determined by type combination and cooking implement
    - Order of ingredients not important
- Quality
-
    - Grade of the ingredients
    - Colour Matching, only applies to 3 item recipes (R-R-R) (R-G-B)
    - Skill with Cooking Implement
    - Level of Cooking Implement
- Buff
-
    - Some recipe provide bonus
    - Require 3 ingredients
    - Require Colour Matching

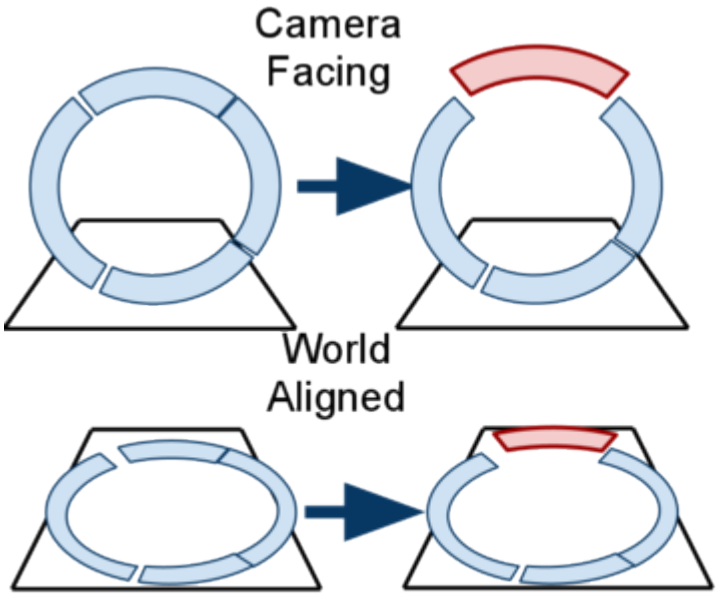
Inserting anything other than the ingredients

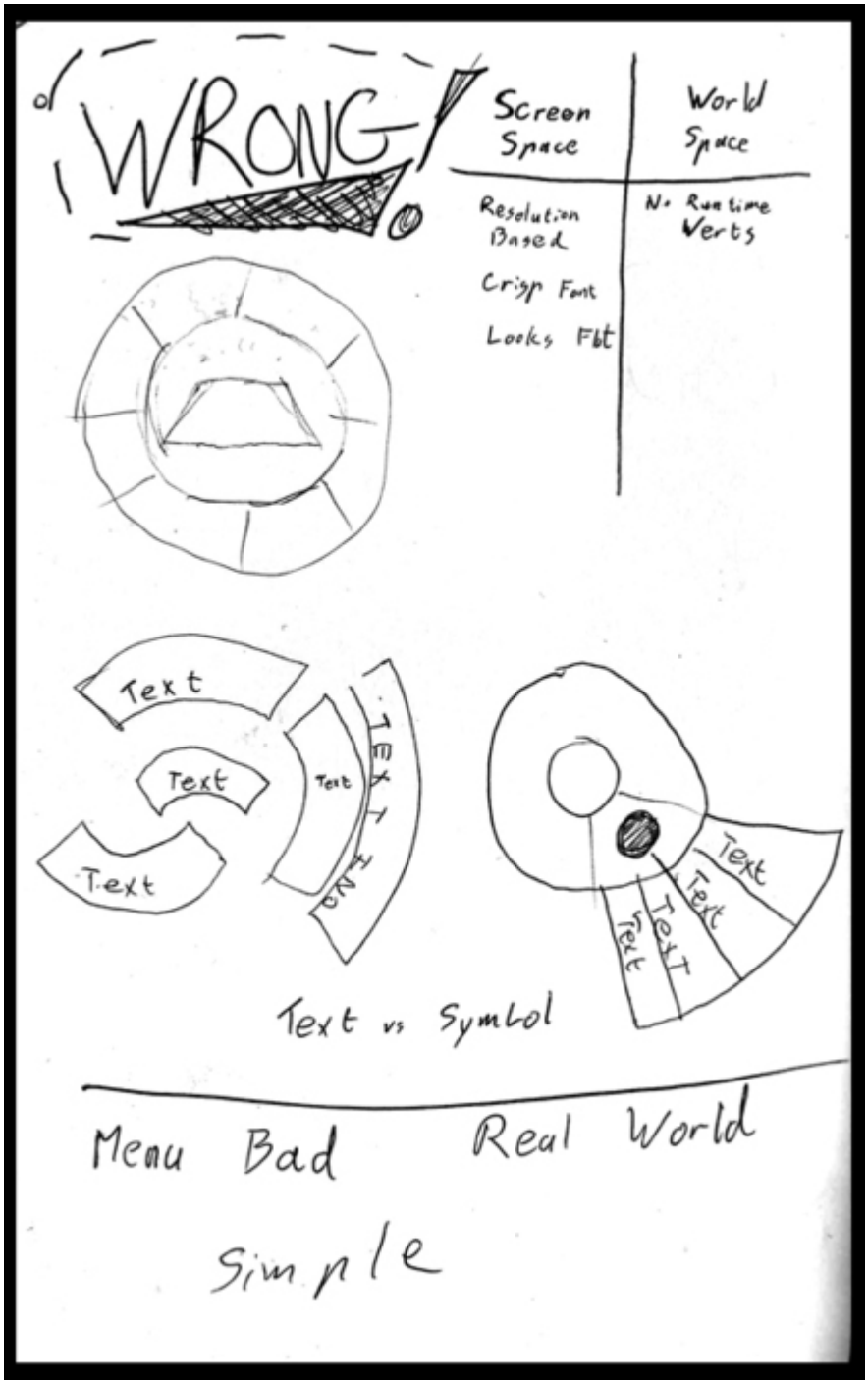
- Produce a mess
- Chance to break Cooking tool
- Destroy ingredients

Interface

A key element of the interface is the menu system

In-Game Interactions





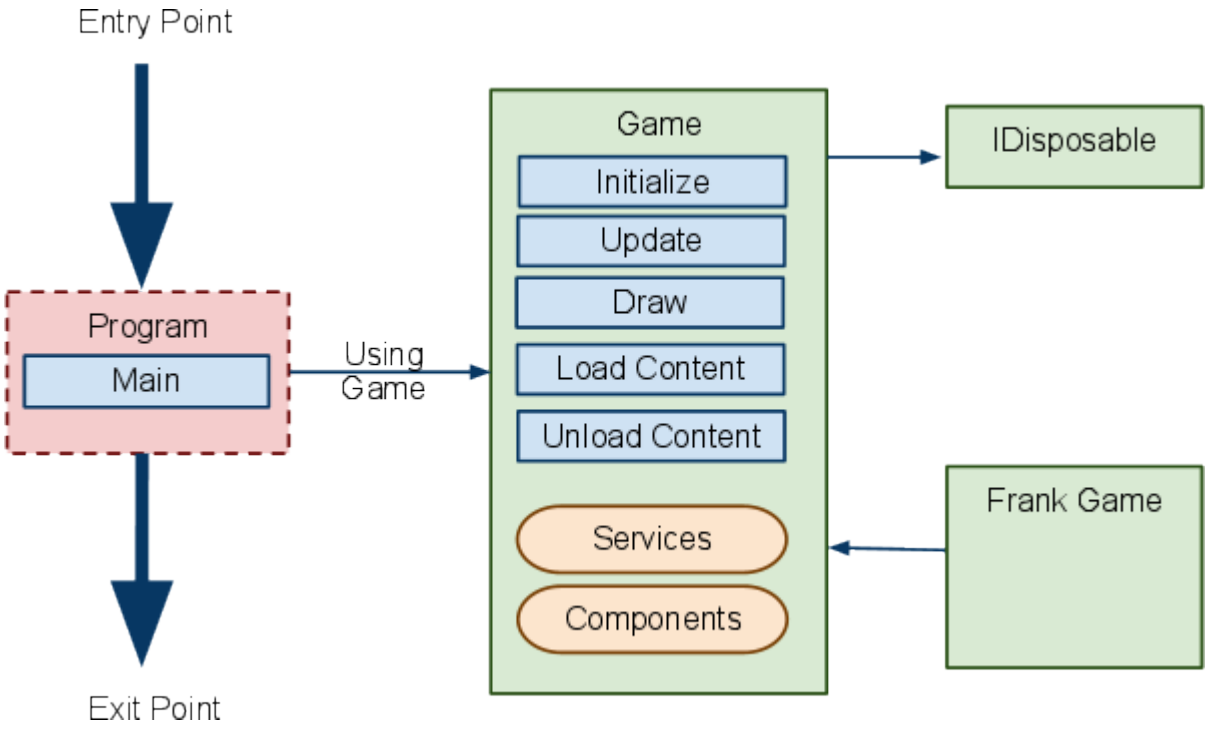
Real World Interactions

After some research I found that younger age demographics react much better to interfaces which are built into the world. Abstract menu and HUD overlays are too much abstraction for younger players. So all interactions were moved into the world.

Thus the introduction of the Blimp Shop.

XNA Game & Components

The starting point for an XNA developer is the Application, Game and Component model. In order to engineer a solution which works with the underlying systems one needs to be aware of their core design principles.



The program entry point is handled by a static wrapper called Program which effectively creates and launches the game. The first thing to be aware of is that we are using C# in a managed environment with Garbage collection. For this reason we inherit the game from IDisposable as that indicates we want the game's memory pool to be cleared out on destruction.

I shall only briefly touch on the game as it has many functions but I'm going to focus on the core feature set.

Initialize

When an item is created in memory we do not want to initalize it as core systems may not be running yet. This also allows us to phase the creation and initialisation of systems. This is vitally important on a console system where memory optimisations play a strong role.

We can setup the game and all key elements at this point. Making sure to set preferences for system level initialisations like the graphics API.

Content Load / Unload

When the game is created and initialised the core systems may not yet be operational, such as the graphics device. For this reason we handle the loading of content or resources in this separate phase.

This also allows us to perform IO heavy tasks at the most optimal time.

Another use-case which evolves from this comes into play more in the component side of things. Some time we want a component or game resident in memory but do not wish the large memory intensive items such as textures and models in memory. In these cases we can simply unload the content and load it again when that component is running again.

This does not occur on the console but on some windows systems an OS level error or event can occur such as the classic lost graphics deviceThe modern implementation of DirectX and XNA no longer suffers this issue, but similar cases do exists. , in these cases we could unload and load the graphics content hopefully saving the game from crashing.

Update & Draw

The update function provides no real surprise, doing what it says on the tin so to speak. The core engineering concept which is enforced here which most hobbyist programmers forget to implement is the separation of logic and drawing.

On consoles and most modern graphics systems we batch process render commands and then package them in a fashion to send to the Graphics pipeline. This decouples the update from the draw and allows the game logic to continue running while the graphics pipeline is processing. When done correctly this means that by the time the frame is finished rendering the new frame would already have been processed and ready to feed to the renderer. Drastically improving performance.

This system however means before the first draw call is made the BeginDraw call is made and after the last call the EndDraw is made. Certain operations can only or cannot operate in this scope. While many other functions cost drastically increase if called in this scope.

One other point to mention is all game components have a render order and update order. This plays a large role in some later components.

Services & Components

Services are a central place to register an interest in a service. This means API for networking, graphics and other services can be registered with. Allowing a game item to react to service level actions. This loose coupling provides a neat engineering solution to distrubted workers which interact with similiar functions.

Components are a way of making atomic elements to a game. These atomic elements can be plugged in as required and interacted. This also makes their draw and update calls independent. A common example used is a fpsFrames Per Second counter. It's a generic component which you only want sometimes and requires no interaction with the game.

Components & Services also remove the need for singletons in many cases.

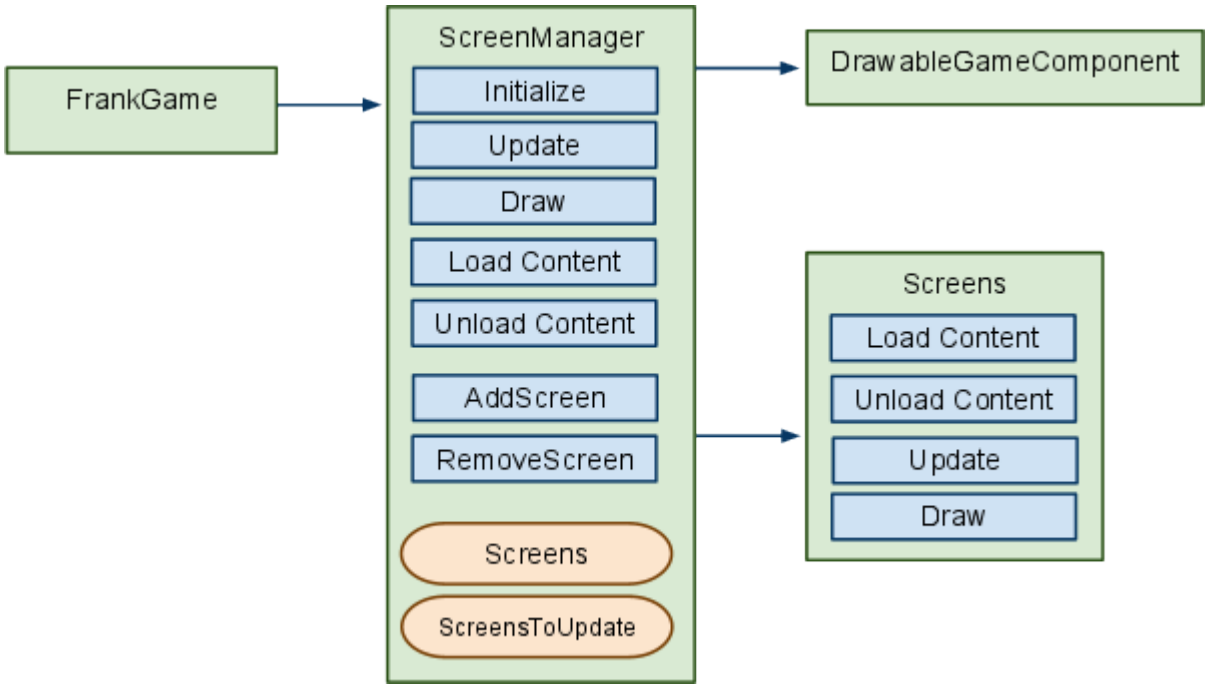
Screen Manager

Inherited from DrawableGameComponent.
This Screen Manager with network states was written by the Microsoft team and I have not altered the system code, but I have written screens for the system.

I chose to use this system as it was a quick way to get a state manager in which supported state management, child states, non-blocking states and interacted with the networking system. Meaning future implementation of multiplayer would be drastically easier.

The ScreenManager is the core element which decides which screens to render, update and provide input to, and in what order. It also handles the transition between screens.

Screens are a simple class. Which wrap state behaviour.



Debug Renderer

Inherited from DrawableGameComponent.

Possible the most important tool is debugging. When working with 3D space a debug render pipeline is required. As the render pipeline is more formalised and draw requests need to be structured in a batch-able fashion.

Problem: Render Order

One problem encountered was ensuring that the render states where correctly setup. For this reason Debug Renderer was moved from is static function calls into a game

component. This allowed us to specify a render order which would ensure it rendered last, with all the correct render states. One limitation which remained was that the debug items could only be rendered from a single View Projection matrix.

Content Pipeline

The pre-processing of content is a good practice in any game with pre-made assests. XNA enforces this with the content pipeline.

Key Benefits

- Standardise data
- Platform Specific
- Reduces Load-Time
- Pre-process
- Invalid Assets
- Code Injection

The game should not need a png, jpeg & gif decoder but instead only a small set of predefined textures. In the case of the Xbox certain hardware level optimisations and limitations need to be applied to these textures. As the data is in a standard platform friendly fashion it is also optimised for fast load times. The platform specific nature means we can also eliminate unsupported formats.

Memory bounds are already specified drastically reducing the buffer over and code injection threats. Which for a controlled environment of a console, or your competitive game are crucial to avoid hacking and cheats. As well as possible unexpected side effects from corrupt data.

We can also handle activities like generating terrain from height maps. Meaning that at runtime the terrain is a static element packaged in an efficient fashion ready for the renderer.

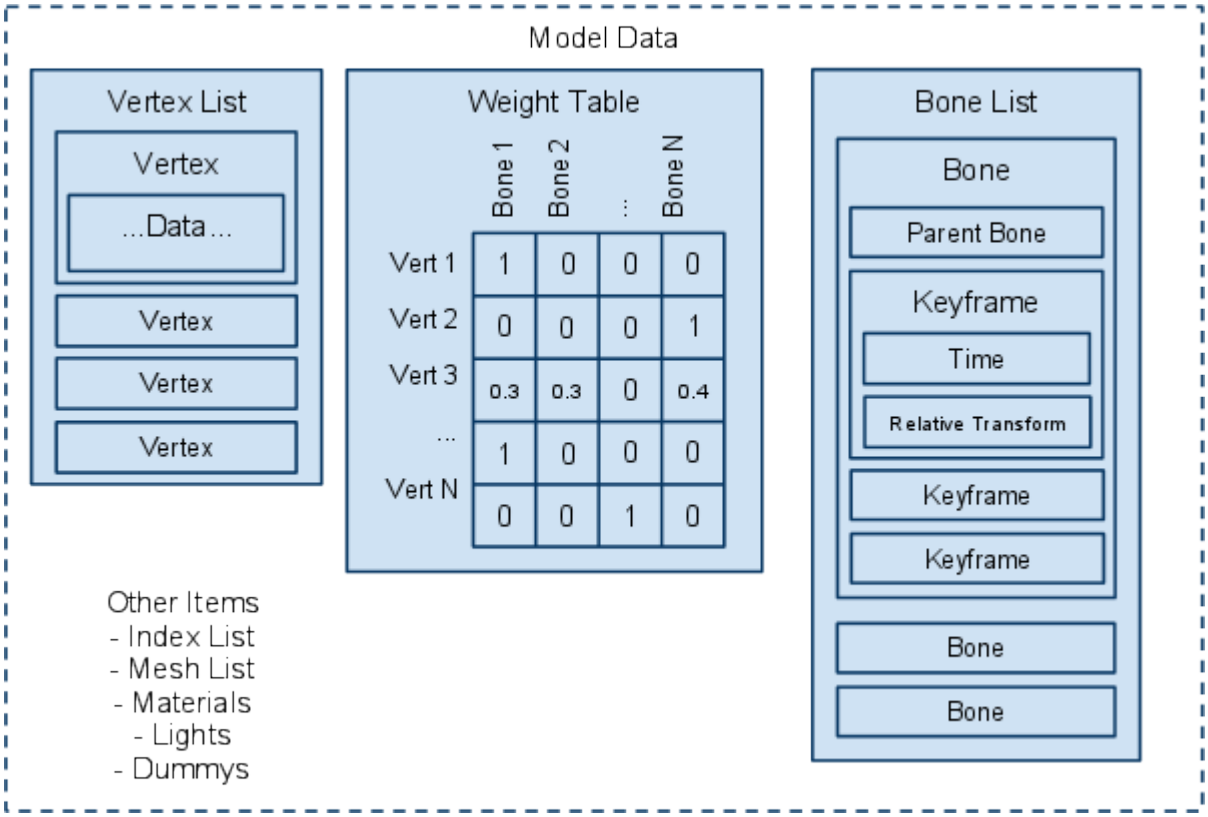
There are many built in processors already written for textures, fonts, model data and XML which we make extensive use of.

Models & Animation

While XNA provides a model processor we required a few things the standard model processor did not provide. For these we needed to expand the pipeline.

Skinned Animation

We took an existing sample for skinned as the basis for this alteration. The system was in two parts processing and playing. I will restrict this section to the processing of the animation data.



We selected FBX as the standard format as it had good support. Now for an animated model we stuck to skinned models with no static elements. So the first task was to eliminate all elements which did not have any weighting on the table. We also needed to normalise the weights so they all added up to 1.0.

As you can see there is a lot of relative data there. The first thing to compress and simplify is the weights table. This can be done by limiting the number of influencing bones to fourIn reality the required number is much lower. In some professionally shipped games I have worked on we have used only a single bone per vert. Multiple bone influences are mostly required for soft items. Such as faces, and muscular joints. , by taking the four strongest influences. We can then take that data and add it to the vertex data along with position, coordinate and other information already stored in there.

We also flatten the transforms on the model mesh parts. FBX exports each mesh as its own component with a relative transform. By traversing the tree we can calculate an absolute transform for each node removing the relative transform.

We then want to do this same flatten process on the skeleton and bones. The key element is that all keyframes will have relative data. The solution to this is called an Inverse Bind Pose. By storing the rest state of the model we can use that to compute the modified positions by multiplying the relative transform by the inverse bind pose.

We can flatten the bones into a list using depth first traversal and store the parent index. This ensures bones will be updated in the correct order without the need for tree traversal and that the data is much more packed, removing references.

Problem: XYZ vs XZY

The world is full of different ways to do things. The classic case is artists like to count from 1 and programmers from 0. Another common issue is that artist applications works

in degrees and XZY configuration instead of XYZ. Thankfully I was able to find an FBX exporter for 3D Studio Max which handled this conversion. Removing the need to implement this in the Content Pipeline.

Problem: Non-Identity Bind Pose

I lost a week of development time to this tricky little issue. It turns out that if a model is skinned and then that skinned model is altered after the skinning is done it generates a funky bind pose. Which of course makes all the animation go screwy. What threw me is many high end importers and viewers compensate for a corrupt bind pose (as best as they can), to minimize the impact on the workflow when an artist makes this mistake.

The error is hidden by matrix math, and when using viewers to test the model I saw it working so assumed the data must be correct. The really confusing thing was my other test models worked fine but the animated hand just would not work correctly. After much debugging, this was the issue which caused me to write a full debug render component, I found a forum post describing this issue.

After starting a new model I discovered the data was at fault. This illustrates the difficulty of isolating the issue in a complex pipeline. It is further complicated when different people, or departments, have responsibility for different sections of the work-flow.

Problem: Multiple Animation Tracks

The FBX exporter that I was able to get for 3D Studio Max does not support the concept of multiple animation tracks. So I needed a way to specify named sections of the animation for later reference. The solution in this case was to expand the content processor. A text-file with the same name as the model is loaded.

The following simple data format was agreed upon.
Animation Name, Start Key, End Key
The processor then split the animation, duplicating the required keys and adjusting the time values of the animation data. Producing multiple named tracks for use in the game.

Custom Effect

With the introduction of shadowing we needed a global effect which could function on all objects. This was also introduced for optimisation reasons. It was in-optimal to remap the model at runtime as it would mean unnecessary material, textures and effects would be processed and included in the data. Not to mention the CPU cycles spent remapping and garbage collecting.

So we went with a texture with global effect model. Remapping most models to use the FrankEffect.

Game Props

Data-Driven is the model to which all things are moving. The concept of the programmer driving every action is outdated and results in code which is not re-usable, and requiring re-compiling to adjust some values. Also as game teams grow larger artist and game designers need to alter logic and behaviour of game elements without getting into the code.

This is the case of the game elements. After looking at the various options I found that XNA has an automatic XML serilisation and that in the most recent version they added Automatic XNB serialization.

Basically if you create a class and mark it up with the correct attribute qualifiers then you can link an XML file to the data. The Content Pipeline can then generate XNBXNA Binary Format files which contain the object data in a fast to read format. There are three basic problems.

1. The class needs to be in a separate project to support reflection which is used to link the data.
2. The .NET reflection API which scans the individual fields and properties only reads public fields by default and needs a lot of help with certain areas.
3. The generation of the reflection list is very slow.

The Props themselves are explained in more detail in the Game Objects section.

Problem: Separate Data

These problems coupled with the polymorphism issue meant that the data class needed to be separated from the runtime classes. The overhead was just two high to maintain a class which would not break the reflection but would also have the required feature set.

The two choices were to separate the data or just write a full custom object pipeline for props. The full pipeline was not a choice due to scheduling so the data was separated.

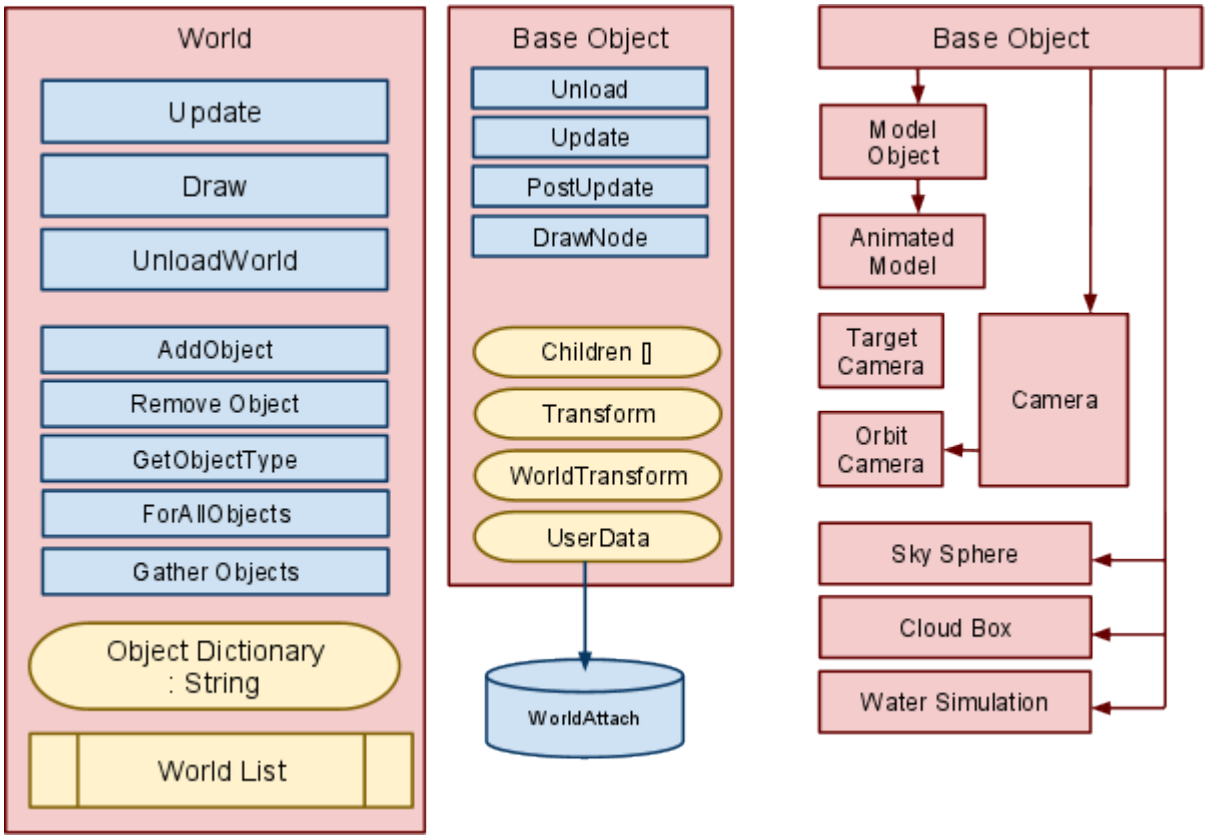
See later for the problem of runtime creation and linkage.

Renderer

The render pipeline is a complex thing to maintain in a 3D game. Performance vs coupling, flexibility vs implementation complexity. There are many proposed methods but not single ideal solution has been found.

Scene Graph

The World Manager was my method of controlling and managing a scene.



World

The world represents one scene. For instance if you walk from outside to inside a building you could store the interior as a separate world. This can also be used to manage what render data is present in the game at any time.

Problem: Special Case Objects

With the introduction of special case items like the sky sphere, cloud box and water simulation we needed to know if these things where in the scene. A consistent naming scheme seemed a silly way to do it.

So using RTTIRuntime Type Information we could determine the type of an object so we could find an object by type. We also added a unique flag which meant we could avoid duplicates. As there is no case where a scene would have two sky spheres for instance.

This solved the naming scheme problem, and ensured duplicates were not added.

Problem: Multiple Render Passes

With the introduction multiple render passes with special requirements for shadows, and environment mapping the render calls needed to be optimised. To this end we wrote a function which would gather all the objects based on based, and depth sort them. This flat array could then be used in the render pass in a more optimal fashion.

Base Object

The base object on which all is built illustrates most the key features of the system.

Unload is a simple, free me up call. The actual removal is handled by the world. This allows for passing Objects between worlds without the need for reloading. Remember there is no actual deletion but instead garbage collection when no active references are present to the object

We have split the Update, Post Update and Render. The reason for this is two fold, neatness and phasing.

Update is where all logic should occur, nothing really fancy.

Post Update handles the render preperation. For instance you will notice we have a world transform and a relative transform. The world transform is updated by the post update. This allows the renderer to not worry about heirachy of items. Optimisations like depth, and effect sorting can then be done. See gather objects.

DrawNode simply handles the rendering of the object.

World Attach

This solved a key issue. We needed some way to cement game objects into the world but obviously we did not want to tightly couple game logic to the World Manager. The solution was the interface technique that C# provides. Any object which inherits from it could then receive updates and notifications from the render system in the correct order.

Model & Animation Object

Model object does a lot of heavy lifting in the project. The majority of render nodes are model data, static or animated. With the introduction of shadowing we needed to introduce the global effect.

This also provided optimisations but at a cost. Each model had to store its texture and assign it as required. Though future room for improvement means the world

Camera

The camera needs to be an object in the world so it can attach to objects and animations.

Target Camera

The target camera simply tracks an object in space.

Obit Camera

The orbit camera can be used as a simple first or third person camera. It is the most common camera people expect to interact with and very useful for debugging. It required a spherical coordinate system.

Problem: Polar & Spherical Coordinates

Originally the camera needed smooth motion which did not suffer gimbal locking, other systems later made use of it. The issue was at first I was using expanded Polar Coordinates which are not really meant to operate in 3D space. I later implemented a Spherical Coordinate helper which could convert between spherical and Cartesian coordinates.

Sky Sphere

As the majority of the game occurs outdoors a sky environment was required. I opted for a Sky Sphere after reading a good blog post by C.Humphrey.

The initial creation of a sphere was simple. The sun was then included based on light direction, using the pixel shader. Tinting of the sky based on time of day was next, also implemented in the pixel shader.

The stars were slightly more complex requiring a cube map to get a good result.

Finally the cherry on the top was the clouds. The key there was the "swirling" which needs to be applied to the clouds and then layered to achieve the best affect. Humphrey had tried using Perlin Noise and other functions but in the end a texture was the solution. His implementation used three cloud maps but was highly in-optimal.

I rewrote the system to use one cloud map and reduced the instruction count to within a reasonable limit. I achieved this by layered the swirls an apply a negative pass. This drastically improved the sky box, especially at night. Once I modified the stars not to shine through the clouds.

Shadows

Shadows are an immensely complex subject.

The first step was to decide on a shadow method. Having researched it before I wanted to try implement shadow mapping using a multipass shadow and shader implementation. In other words volumetric shadows but not using the depth buffer.

What you do is first you gather all shadow casters. You then calculate the scene bounds. This was a tricky step. The camera is not necessarily looking at the centre of the scene. So you get the bounds of all casters then expand it based on the view offset.

This gives you a light view projection matrix, which you then use to render all casters but using no textures or lighting. You save the depth to a render target. The target uses F32 or Single Float model to ensure maximum accuracy. Also because casters tend to be closer to the camera and the way floats are optimized it's best to subtract the result from 1. All results must be normalised as it's a depth buffer.

The result was very blocky and ugly. This was due to the render target being spread over all casters. There were some tweaks that could be done but I needed something drastic.

At first I tried smoothing the shadows using PCFPercentage Filtering which uses a Poisson DiskA noise generation method for evenly distributed points. to sample multiple nearby points and then smooth the result. The result was better but looked fake.

After much research in the games gems & gpu gems book I found one article posted by nVidia. Percentage-Closer Soft Shadows by Randima Fernando, abbreviated as PCSS.

PCSS basically does two pass test. The first sample test looks for any blockers between you and the light source. It does that by using Poisson Disks to sample the depth of the blockers, returning the amount of blockers and the average depth.

Assuming there is at least one blocker it then calculates the penumbra size based on the light source width, and numbers of signals blocked, and average distance of blockers. It then does the same PCF filtering I was doing previously but scales the size of the disk based on the penumbra.

Meaning that items close to the shadow caster get defined shadow while distant objects cast a soft blurred shadow. Extremely distance objects cast almost no shadow. I have still not quite got the numbers right but the method is in place.

Cloud Box

The sky sphere method is designed only to be seen in one hemisphere in a game. As there is normally a horizon in most games. So I needed a way of boxing off the scene and creating a horizon.

My first attempt was to use a cloud box, reusing some ideas from the Sky Sphere.

Knowing I had limited art resources I looked at using a flat plane and then trying to simulate depth by projection down the clouds and creating shadows. The lack of a physical depth meant close up however illusion was horrible, and as the 360 does not have shader model 4.0 I could not alter the depth in the pixel shader based on the cloud map.

At the end of the day I deemed it inadequate and moved to the water solution. The system would have worked if a cloud mesh had been created by an artist or a meta particle simulation was used.

Water

If the island wasn't floating in the clouds the next logical choice was the ocean. This also solved the need for a fishing pond in the game. Frank could go sea fishing.

A fluid simulation was quickly ruled out, and after some research a height-map implementation of water was found. It gives fairly good results for large bodies of water. It cannot fold in on itself or simulate fluid dynamics but for a body of water which was mostly flat it worked well.

The simulation works like this. Have three height map buffers, current, previous and final. If you want to alter the water you alter the current height map. You only update the water simulation every X milliseconds. At all other times you just lerp between previous and current buffers and save it to the final. This gives the smooth transition quality and flow, which makes it look much nicer.

Each update of the simulation you simple sample the surrounding points on the current buffer and average it out, applying a dampening factor to account for loss of energy. You then swap the buffers.

The buoyancy and density were a really treat. Turns out if you can assume a plane (with modified heights) and you want to test the buoyancy the computation is quite easy. You simply need to sample every nearby point covered by the sphere, get the relative depth calculate the buoyant force which is always opposite and relative to gravity, meaning it's always up.

Displacement can be done by setting the height and running a two pass calculating the downward force and buoyancy force.

The graphical mesh of the water needed to be made at runtime to order. The physical and graphics dimension need not be the same. This allows to balance between CPU & GPU load. It should be noted there is a slight performance boost if they are the same.

Environment Mapping

In order to make the water look good Dynamic Environment maps needed to be created. The logical choice was a cube map as it's supported at a hardware level and gives high quality results.

In order to do a cube map you need to render the scene six times (once for each face of a cube), which is a large cost but can be done at a smaller size. A further optimisations was to only render the skysphere as that was the bulk of what was reflected in the environment map.

The next problem is a cube-map is only accurate for a point directly at the centre. For all other points you need to fake it. You do this by getting the reflection vector and projecting it in a virtual object.

There are two methods. The more accurate and expensive method is to project a ray from a point and get the intersection with six planes. Testing which looks the best candidate. The other way is to assume a virtual sphere instead of a cube. You know the intersection point is going to be at X distance from the centre point. With some basic algebra and the ever loved and optimal dot product you can get a distance, modified by position giving you a modified normal.

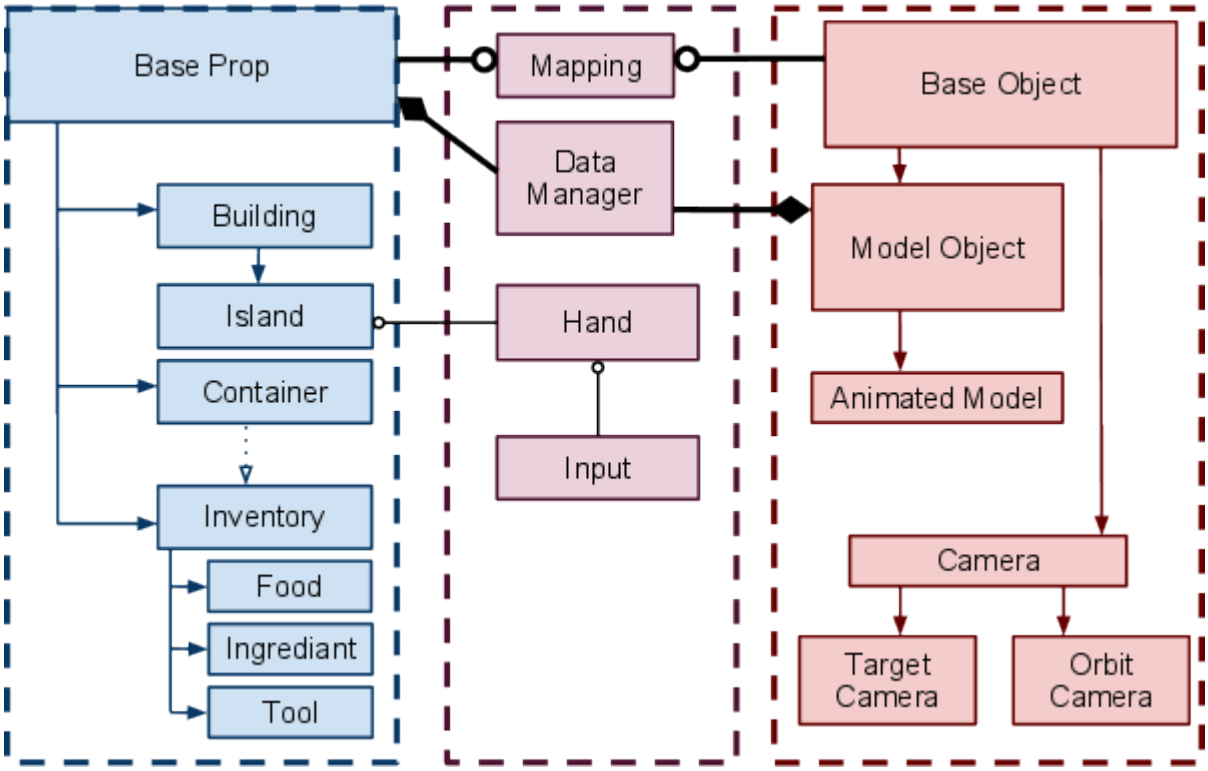
Game Objects

Data Manager

The first problem as referenced to earlier was that the data needed to be generated as settings and object created from those settings.

To that end we created the Data Manager. This class contained all the possible objects and their resources loaded. The model is that you query a the manager for a new instance of a object, the manager also deletes objects.

Decoupling



The Data Manager was crucial to creating and managing objects. We also added a mapping to the objects.

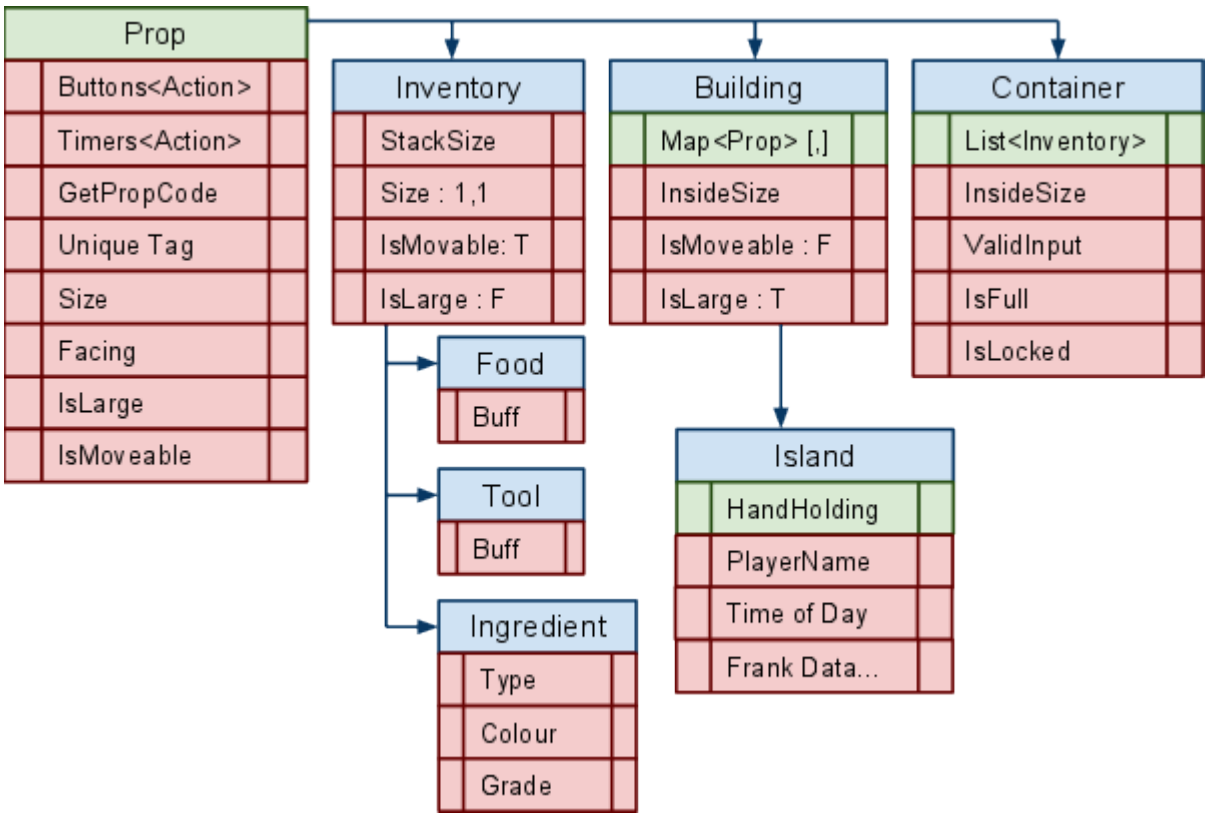
A prop had a reference to it's Render Model, which it did not need. See the Pipeline and World for persistent objects with no render resources. This allowed it to inform it's render model of updates.

The base object has a reference to it's prop through the world attach system which is given to it on creation by the data manager. This loose coupling solved many issues while remaining fairly flexible.

Problem: Massive Switch

One issue is that the data manager currently has a huge switch statement. It should be replaced by a Factory pattern. I could not find a game object factory implementation in C# sharp however as all the patterns I know use C++ trickery which do not work with managed code.

Game Prop



Game prop's being defined mostly by data was a massive performance increase.

Building & Island

Originally the Island was a special case. After looking at the coupling and linkage to the World Manager system I realised that any method in which Island was a special case would require tight coupling.

I also realised that I was duplicating lots of functionality in Building and Island. The system made a lot more sense after re-factoring it so Island was inherited from Building. Solving all the coupling issues I was struggling with.

Button & Timer

The Buttons and Timer are key data driven element.

At first I was looking at a labelled delegate system. But too many limitations came up. For instance the desire for multiple actions or specifying an argument in the data.

After several version with delegate libraries I realised a more flexible system was needed.

- Processed at Runtime
- Varible Arguments needed
- Multiple actions in one command

The solution was to store commands as strings which would be tokenized and parsed at runtime. Creating a simple runtime scripting system. Requiring very little effort to add new commands.

Example

> *Command_Name.Arg1.Arg2.ArgN; Command_Name;*
> *Command_Name.Arg1.Arg2;*

One key element was passing in the caller, so the command had a reference point to the world from which to act.

AI

Not yet Implemented

Networking

Not yet Implemented

Progress Log

Tumblr Log

This log provided the motivation and publication spoken about in Project Management sections.

Case Project

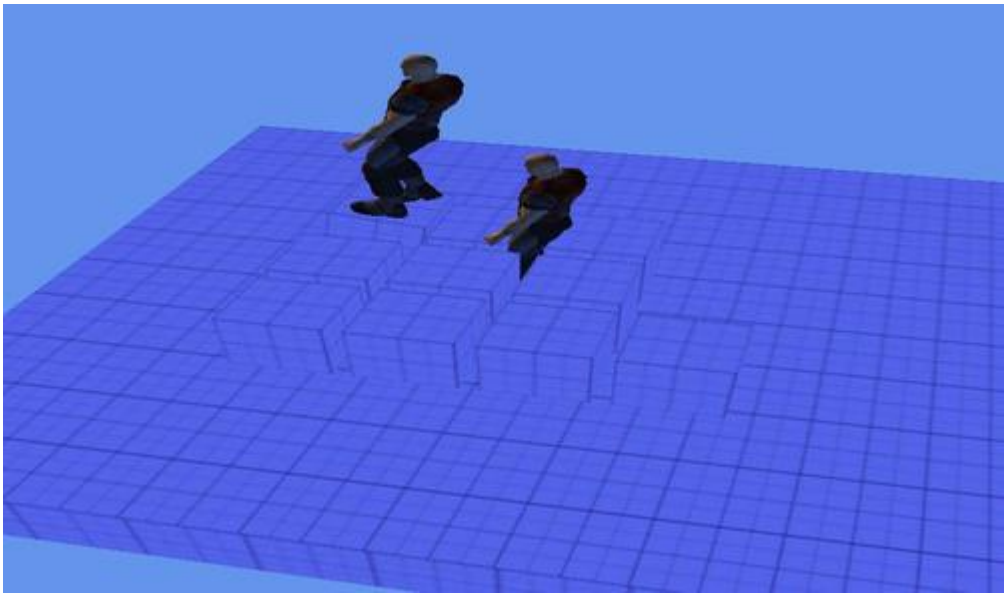
1. Created Project
2. Menu System
3. Networking
4. Skinned Animated Models

Sounds like a lot but in reality I stole the Network State Sample and Skinned Model Sample used some duct tape and got to work. I have no shame because Im not a tech junkie and the quicker I can get to the game the better.
Onwards to fixing the render glitches& can you see them ;)

Getting into C# and Feeling Stupid
Todays Goals

- Build World Manager
- Make base Hand & Frank
- Get Hand to Pick up Frank

So currently Im doing one of the things I like least. Building low-level systems and technical plumbing. General I like to work one of two ways. On the bone raw apis dirty and fast, or working on a high level system.
The building of engines and low-level framework style system is a crucial element which I never enjoy doing. Also my unfamiliarity with C# and XNA is a big slow down.
Transforms: Target or Moving?
The question has arisen about whether target transform or moving transform is better. Ive used both system in the past.
Target Transform: Lerps between previous and target to return the current transform.
Moving Transform: A relative transform which is applied to current transform.



Small Progress but still progress
Okay so doesnt look very different.

- Multiple Animators
- World & Object Management
- Instanced Objects & Animations

The World & Object management took me so much longer than it should have. I had to wrap my head around methods of garbage collection that C# uses. Ah well lets see if I can get some basic object interaction in.

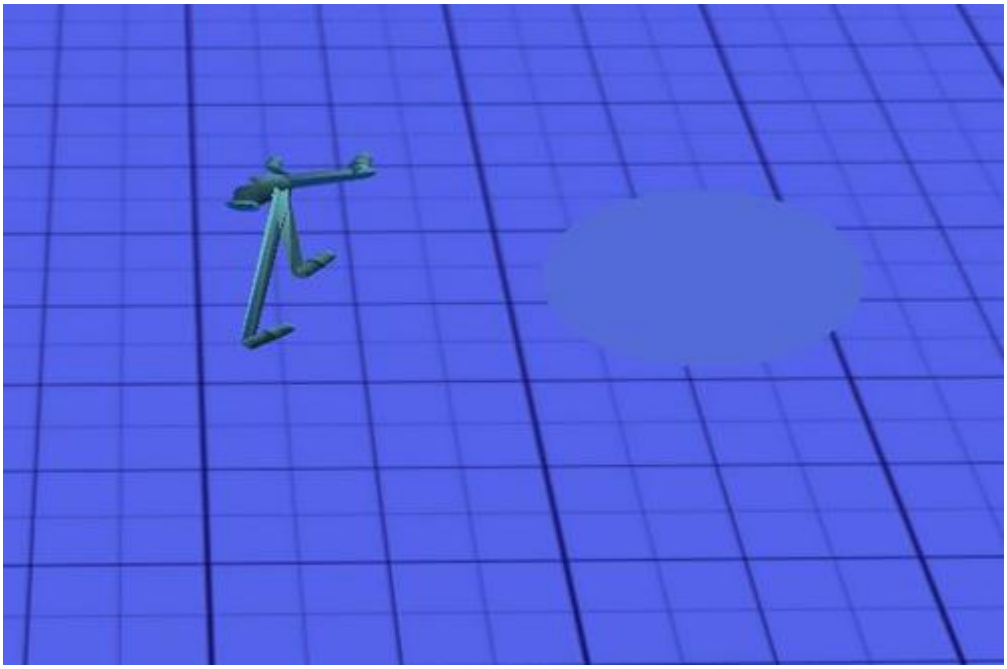
Splitting Animations
The last roadblock I hit was tagging animations for export. Well after talking to the artists at work no-one knows how to do it in 3ds Max 2009 (or any version of Maya). So after some digging I found this post on the XNA forums.
So by parsing a separate text file using the Content Pipeline I can export and tag animations. So a bit of shuffle work and bada boom, problem solved.
Sorry no visual progress to show, but its a technical issue solved which makes me happy.

The Hand of&.. Me
So now that I have the multiple animation system in place I need something to test it with. So the key things in the game are Frank & The hand. So I quickly threw together a hand which does the following.

- Idle
- Pick Up
- Idle Holding
- Put Down
- Drop
- Pet
- Tell Off

So now that's done I need to use the dummy (the ball) and attach an object to that so the animation of the picked up object matches the motion of the hand.
So next steps

- Hand in Game
- Ball in Game
- Move Hand
- Play Separate Animations based on Input
- Location Based Picking
- Dummy Attachment



Brick wall.
Importing really simple models with a low bone count with no issue. The hand has been glitching all day. Still no closer to solver the problem. Unfortunately Im working with Visual Express C# and my old Visual Studio Pro 2005. Which makes debugging the pipeline a bit of a bitch, as Visual Express does not have a JIT Debugger.
Not sure what part of the hand is given trouble, to be clear other simplier skinned test models work fine.
Any ideas?

Right Ive moved over most the nice LD code into the Frank project, cleaned up a few things here and there. Made it play nice with the World Manager system.
First thing I did after getting the camera controls in was apply the debug renderer to the stupid animation problem Im having. Now to clarify both the Worm and Hand have translation and scale applied. In every way other than complexity they are the same.
As you see the hand bones are in the correct place. What you are seeing are the World Transforms which then are multiplied by the Inverse Bind Pose before being based to the shader. So the job is to narrow it down as follows, Ive eliminated the~~strike-out~~ words.

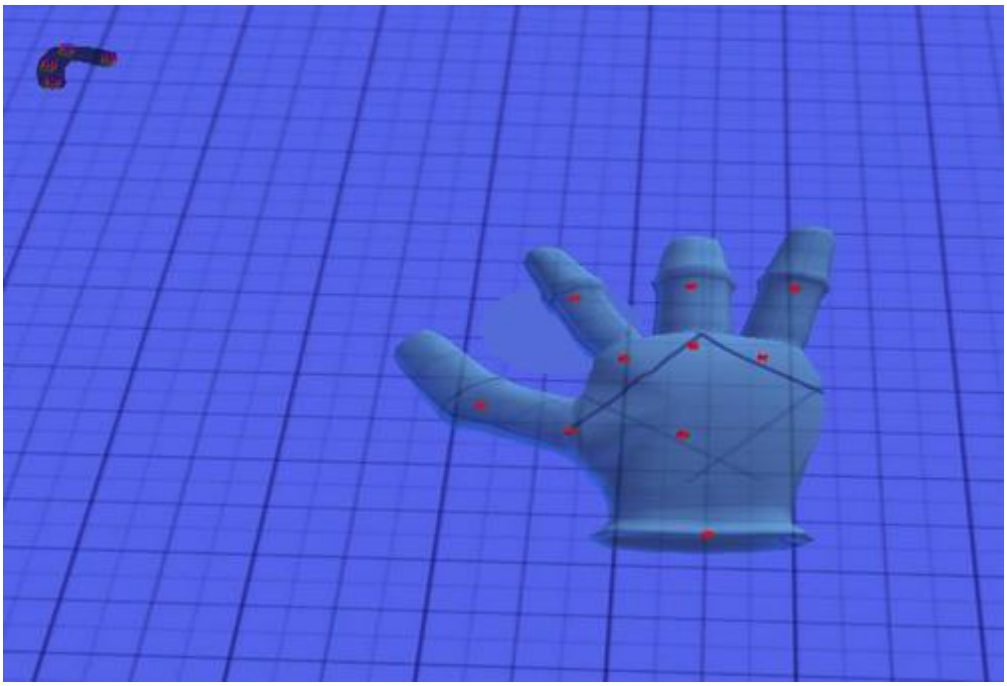
- Data
-
    - Art in Max
    - Exportor
- Code
-
    - Content Processor
    - Setup of Bind Pose & Inverse
    - ~~Animation of Bones~~
    - ~~Computation of World Transforms~~
- Shader

SVN Comments below which sums it up

- SphereHelper.cs is a Spherical Coordinate Helper class
- PolarHelper.cs does the same for Polar Coords
- A DebugRenderer component makes life easier
- New Cameras
-
    - OrbitCamera for 1st & 3rd person Sphere Camera
    - TargetCamera for tracking a node from another node

- General clean up in GameplayScreen
- Moved GameplayScreen input helper functions to InputState
- Added Some input helpers to InputState
- Cleaned up Debug Camera controls

So any clues on this animation bug?



Stupid Artist!!
Oh no wait& thats me.
As far back as Wednesday last week when I was working my way through this problem I joked about the artist being stupid. This was a sort of anti-joke because I get annoyed when a programmer doesnt take responsibility for a bug and they multiple re-assign occurs. So much so Ive developed a blind spot to compensate and assume the artist work is correct.
In my defence it turns out many importers and viewers actually have hacks and fixes in for common artist error. So the hand model looked FINE in the official viewer.
Turns out I had rigged it wrong and as a result the bind poses couldnt be calculated correctly.
Solution: Re-rig and animate the model
On a side note Ive lost easily 20+ hours on this single issue.

More notes on Game-Play
So Ive been at Uni and running around town so most of my work was in my note-book but it gave me some time to form some thoughts, and define some game-play. Also after much thought a stunning thought came to me. **NO VIOLENCE!!!**

Player Actions

- Frank
-
    - Pet / Scold
    - Feed Frank
    - Move Frank
    - Highlight Point
- Prop

-
    - Move
    - Harvest
    - Interact
- Start Game
- Buy Prop

Props

- Bought with Coin
- Upgraded with Coin
- Can be either Exterior or Interior Item
- Production
-
    - Food
    - Coin
    - Time-Based
    - Max Capacity
    - Does not Spoil

Buildings

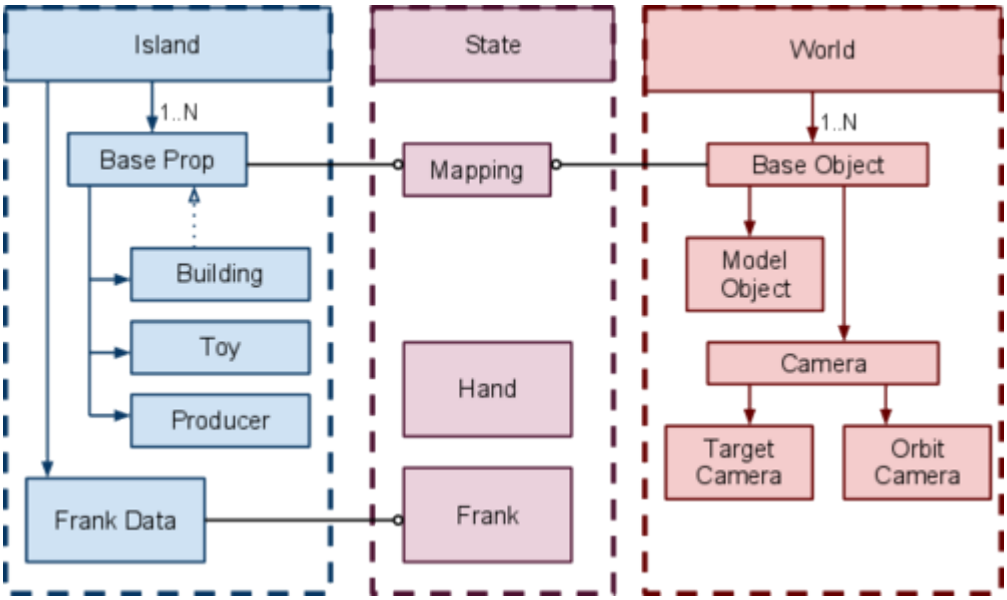
- Bigger on the inside
- Can be upgraded with Coin

Mini-Games

- Practice Mode
-
    - Teach Frank Skills
- Daily Tournament
-
    - Coin can be Won
- Race
-
    - Get from A-B in shortest time
    - Hurdles and Path Finding
- Treasure Hunt
-
    - Time Limit
    - Find X items in time limit
    - Find hidden items for bonus points
- Tag
-
    - Time Limit
    - Multiplayer only
    - One Pet is it
    - The others need to run or hide
    - On touch the target becomes it

Frank

- Appearance
-
    - Pattern
    - Colour
- Stats
-
    - Speed
    - Jump
    - Brain
- Skills



Coupling and Forethought
So last night I started looking at setting up the game elements and save data. The moment I started I saw coupling issues rise up like the ugly beast.
Thankfully Ive seen this monster and I know how to deal with it. I went to bed with pen & paper. This is a problem which is not solved by sitting at an IDE but instead with pen & paper and talking. Too often have I seen projects with *terrible* coupling or abuse of const and other coding sins.
So after some thought I think I have a structure which holds up under scrutiny.

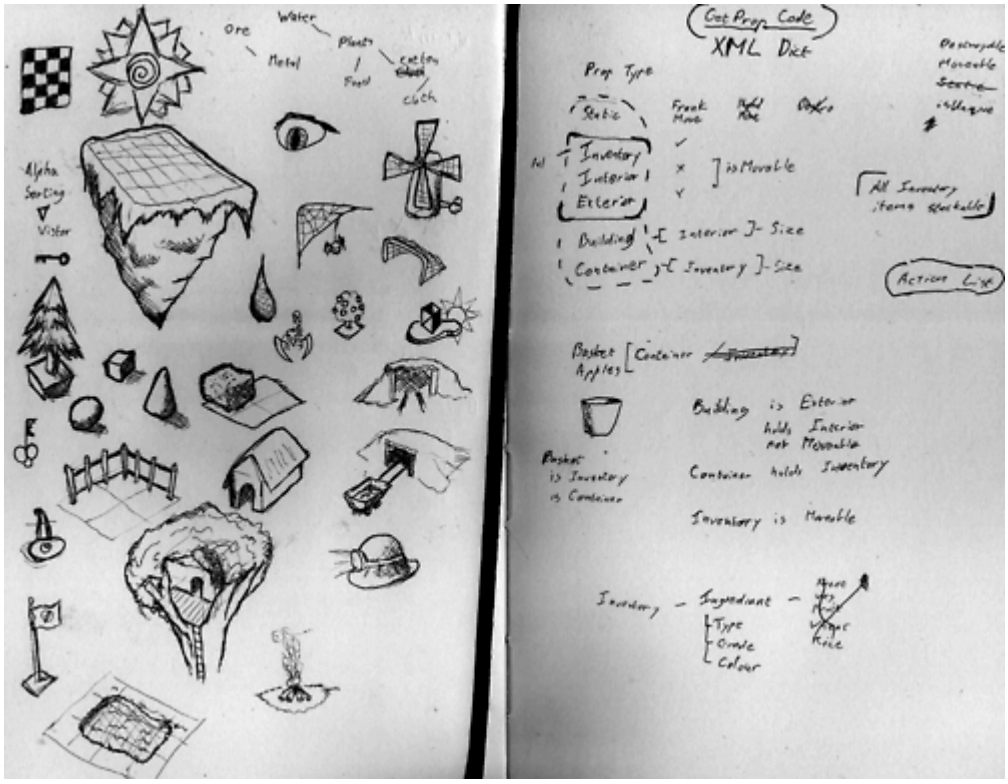
XML is XtreMLy Tasty Food
So after the coupling issue Ive been looking at Object Factories and inventories. Now something I learn to love more and more is **Data-Driven** development. I cannot express my love for it highly enough.
Also thanks to the amazing work of Shawn Hargreaves and all the XNA folks we have automated reflection and xml processing. Which let me tell you is just MAGIC!
The problem Ive been battling with the last few hours (other than sheer amazement at how awesome the XNA system is) is defining the data structure and whether the class itself should be spat out or a data format, or settings style system is more appropriate.
So the solution as I have come to found a lot recently is use case. Before you build the system, write a scenario, set of rules, or list of items. Then use those to drive the devolpment.

As I often have said to the team on our new project at work, which is on a very tight schedule, make a choice and be definite. Its okay to be wrong, but imprecise and fluffy is unforgivable.

Use cases, defining more stuff.
Well I suggest following **the link** for loads of stuff but the biggest thing is cooking. I knew some simple things were core to the game

- Time Based Interactions - Farming
- Activities - Fishing, Fairy, Milling
- Coins & Customisation
- Mini-Games - Race, Egg Hunt, Hide & Seek

Now the one thing that I keep coming back to is the low-level of player interaction. I just need to keep reminding myself that the key element is **Franks Autonomy**. The player can encourage, scold, and highlight. Not to mention choosing how to spend coin and decorate. Ultimately their role is a passive one.
The scope is important as well as it needs to be within reason. The cooking scared me at first till I realised that I could easily just re-use images and use text to describe things. I have a few other ways in mind to drop workload. The final scope will be defined by Friday. Im also hoping that I will have enough to fully write the xml data format, resources, and skeleton classes.



Updated the document from my notes, and a few use cases.
Clean Version Here
The main break throughs and choices are below
Prop Code
All the prop data will be created from an XML file and parsed into a dictionary which is indexed by Prop Code. All new props must be created from either Cloning or using a Prop Code to create an instance
Unique Tag
This means only one instance of any prop using this tag exist. This solves a lot of problems with upgrade items and tools and such. Not to mention the main building
Timers & Buttons

Possible the biggest break through was the throwing away of a production prop sub class and moving to a timer and button system. Basically when the button is pushed by Frank or the Timer expires a delegate is called which is assigned via data.
Using this system, and the concept of morphing into a new object we can model trees growing, ovens and a few other things. The other attraction is Frank can examine objects based on Timers and Buttons. Which lays nicely into some AI reactions I have planned.

The Rest&

Loads of small things I realised and a few things Ive made note of edge cases and the like. For instance containers ValidInput, and IsFull solves quite a few problems.
I have more to say but Im going to leave it there.

I really wish I had a pretty picture but I dont. Sadly its a problem with grunt coding tasks is there is very little pretty to show for it. In fact most of my work today has been in the Content Pipeline which runs at build time. So its not even runtime code.

XML Data

The XNB system in XNA is great but I had a few hiccups with it. I still dont think Im using it in the best way possible. However after ripping up tons of pages of notes Ive gone with the best solution I can currently think of which no doubt has loads of problems, besides some of the ones Im already seeing.

Model View Controller

Soooo sooo easy to be a naughty coder. However as this is an academic project and I can be slightly OCD even with production code Im following it as MUCH as I can. The controller is slightly too coupled to the state which it controls but de-coupling is more messy.
Always always realise the so called rules of software engineering are generalisations and guidelines. You should ***never*** break a rule without first understanding it. They need to be flexible and you need to be flexible because the key factor which makes someone an engineer is a way of thinking, **analytical problem solving**.
A decent education in engineering is mostly about how to think. The tools you use are easily learnt but its the process and thought model which is hardest. I think thats why you see so many true engineers have cross disciplines.

Tomorrows Tasks

Well now that a lot of grunt work is in place. The goal for tomorrow is to have the following working.

- Tree Prop - Loaded from Data, Go through full life cycle and interaction
- Oven Prop - Loaded from Data, Go through full life cycle and interaction

Bigger than it sounds :)

AAAARGH!!!!
Really slow progress, here are some of the hurdles Ive been hitting.

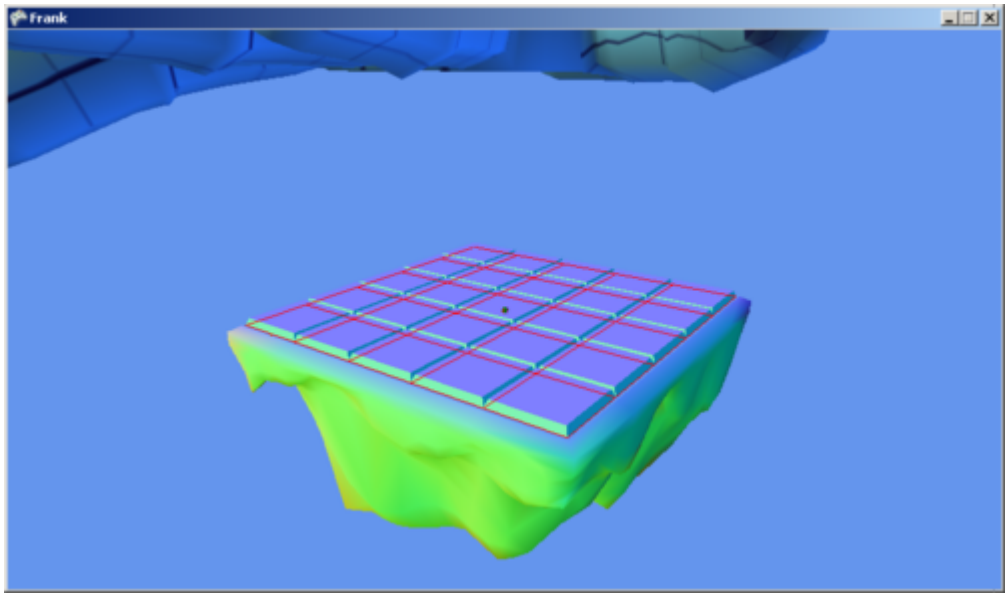
- Baking Shadow Maps in 3D Studio Max while maintaining the UV (a bad tangent)
- Confusion between ActiveTransform and Transform. Renamed them to WorldTransform and RelativeTransform.
- Broken input needed fixing
- A few confusions between data and active
- Some more confusions between keeping the units and such in order

Ah well to bed :(

Hitting a massive break trying to get Model-View-Controller to work with Data Driven model.
Ive found out some nice stuff about C# activator but it gets messy.
Ultimately I think Im going to have to through a few engineering scruples out the window and hack it if I want to stay on track. Which is IMMENSELY frustrating. Going to sleep on it and see how I feel in the morning.

Data Driven Objects
Ive come up against this problem in MANY projects and to be honest never found a solution Ive liked. The problem is you dont want to hand-code each unit and ability ect& you want it to be data driven but this is quite a task.
Some of my biggest personal projects have ultimately died/failed due to this issue. Including my South African final uni project, my big Descent clone and my X-Com style game, oh and that hacking game. I REALLY need to find a good solution to this.
Sadly Ive never attacked this problem in a work environment. All the articles Ive read on it are vague or focus on a tiny part of the problem. I think I might need to look at the game gems books, and do some more research. I hate how much time Ive lost on this issue which Ive seen so many times before.

Output Log
Created World[Game World] Building Basic Prop [Template] Frank.Island.WorldProp Building Basic Prop [Building Template] Frank.Island.Building Building Container Prop [Container Template] Frank.Island.Container Building Inventory Prop [Inventory Template] Frank.Island.Inventory Building Ingediant Inventory Prop [Ingrediant Template] Frank.Island.Ingrediant Building Tool Inventory Prop [Tool Template] Frank.Island.Tool Building Food Inventory Prop [Food Template] Frank.Island.Food Added Obj[Island] to World[Game World] Added Obj[Hand] to World[Game World] Added Obj[HandCam] to World[Game World]
It might not look like much but that little but of text output proves the hacked type system works. Its horrible, sadly Ive wasted a weekend trying to do things the correct way and failing.
If I have time I want to investigate alternatives
// Horrible HORRIBLE System!!!
// Possibly replace if time permits, or else gourge out my eyes in shame
// Game Gems 1: Magic of Data-Driven Design by Steve Rabin
// Game Gems 2: A Game Entity Factory by F D Laramee
//
// Reflection and C# System.Activator.CreateInstance(&)
// See: http://forums.xna.com/forums/t/39891.aspx
Ah well it works and I have a meeting with my supervisor. Hopefully by tonight I will have that Tree & Oven in game.



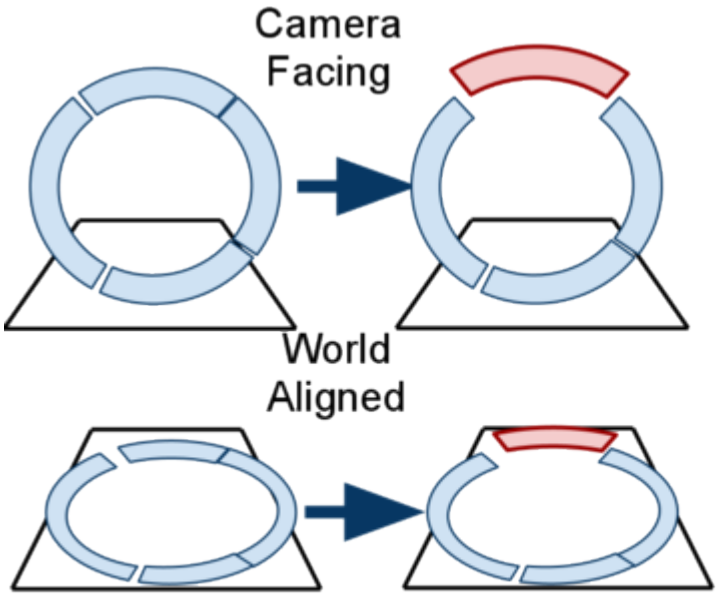
Island Scale

Staying late at work to do university work is REALLY awesome, and so works. My work setup is just so much better for coding.

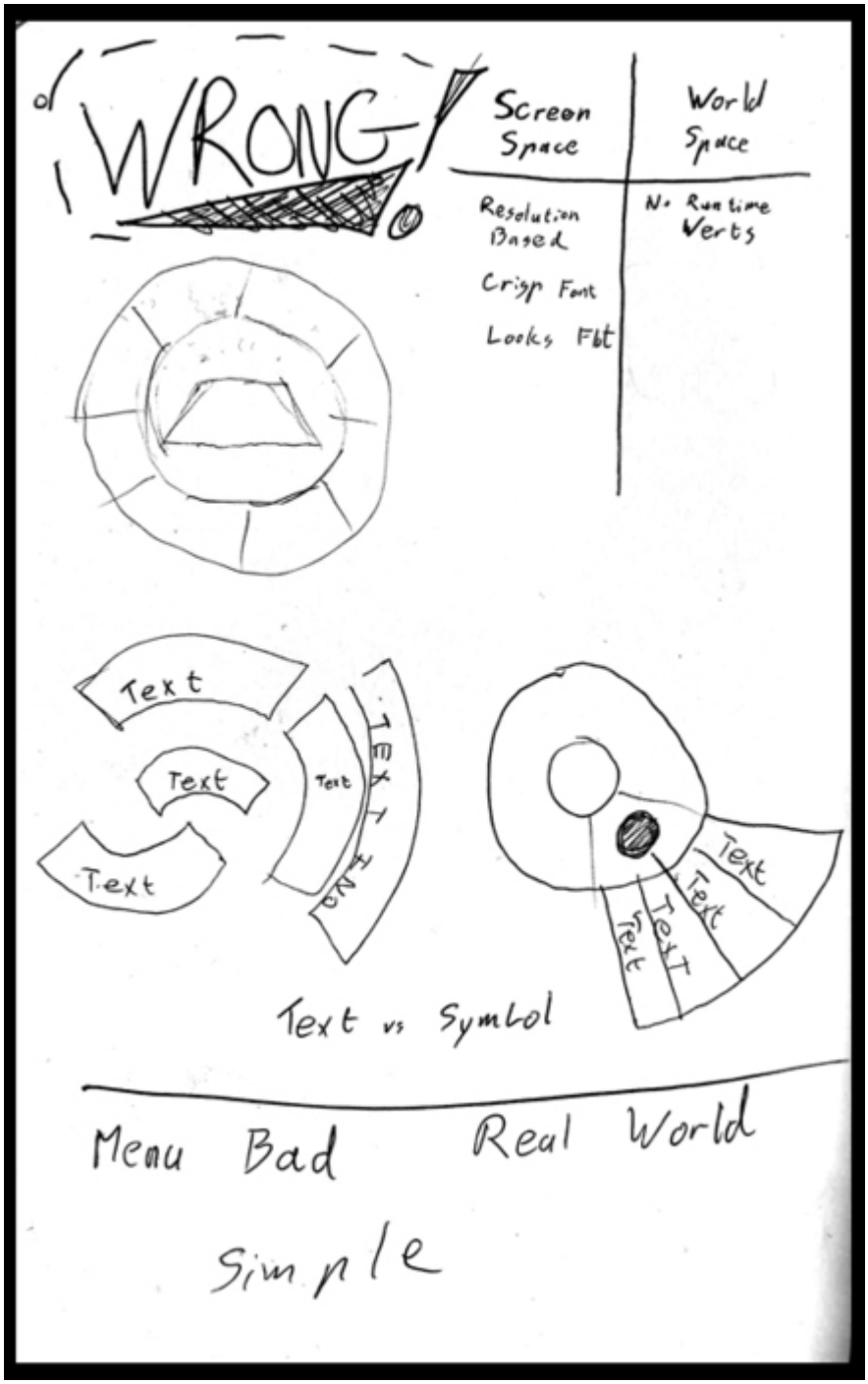
- Pro vs Express
- Two Screens desktop vs Laptop
- Work vs Bedroom

Anyway, getting the island render stuff in. Sadly I need to catch a bus home but Im hoping to crack on with purchase & place items.

So no progress yesterday as it took me forever to vote, and then I got lost in endless election cover and got no sleep, zombied through work today.
Anyway got back and looking mostly at setting up the menu and gui system for buying and setting up stuff. Some nice progress but no pretty pictures yet.
I was hoping to have pictures tonight but lost so much sleep need to catch an early night. Pretty UI tomorrow.



GUI Work
The big choices Im facing at the moment and playing around is on two fronts, GUI Behaviour and Render Pipeline.
Fighting through some choices& will post more shortly



My Kingdom for a Team

Whats the best tool to explore and solve problems? Explain it to someone else. Its good enough for Sherlock Holmes and its good enough for me. I hate working in isolation because you dont need to justify or explain yourself to people. So massive gaping holes form in your logic.

Long story short is explain in the picture. WRONG! It took me a few experiments to realise this but I finally did.

1. Radial Menus are conceptually complex
2. This makes them good to manage large amounts of choices
3. ONE BUTTON!!!

Okay this game is focused on being simple and broad age range according to the brief. So one button is good. This combined with the mantra, *You can do what Frank can do* . A break through moment I had on the bus which confused many people last week.

So whenever you push the button on something you activate the action Frank would. The only additional actions are, Highlight, Pick-up (optional) & Customize(optional). When interacting with Frank the actions are Praise / Scold, and Pick-up.
Now there are other actions. Like looking at stats, buying things from the shop, selling things, ect& But I realised I should bake these into the world as objects. That way they are easier to identify and interact with (kids prefer less abstraction). They are self-contextual and tbh look cooler.

The Other Problem

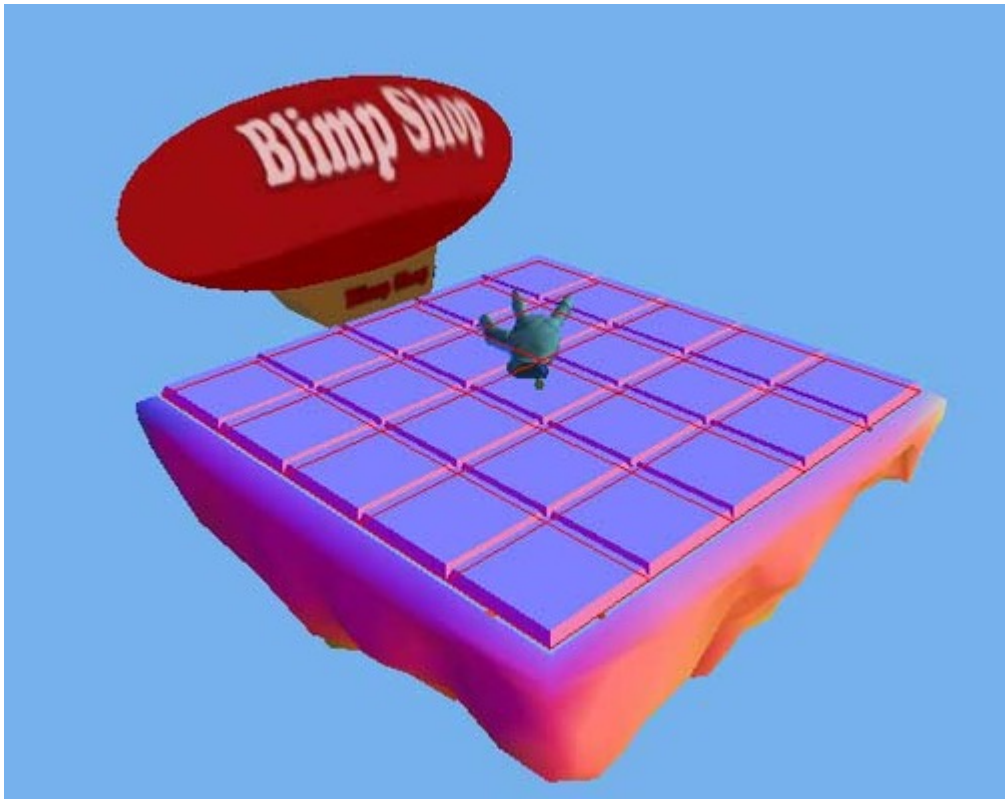
So what have I been wrestling with all day other than my own stupidity. Well a design choice and some coding issues.
The design choices involve things like.

- Camera Facing / World Orientated (see last post)
- Facing: Cleaner Smoother Fonts (made less of an issue by HD)
- World: Feels more part of the world
- World: Able to animate in 3D and look cooler (sounds stupid but cool is a test)
- Floating above, fixed location or surrounding

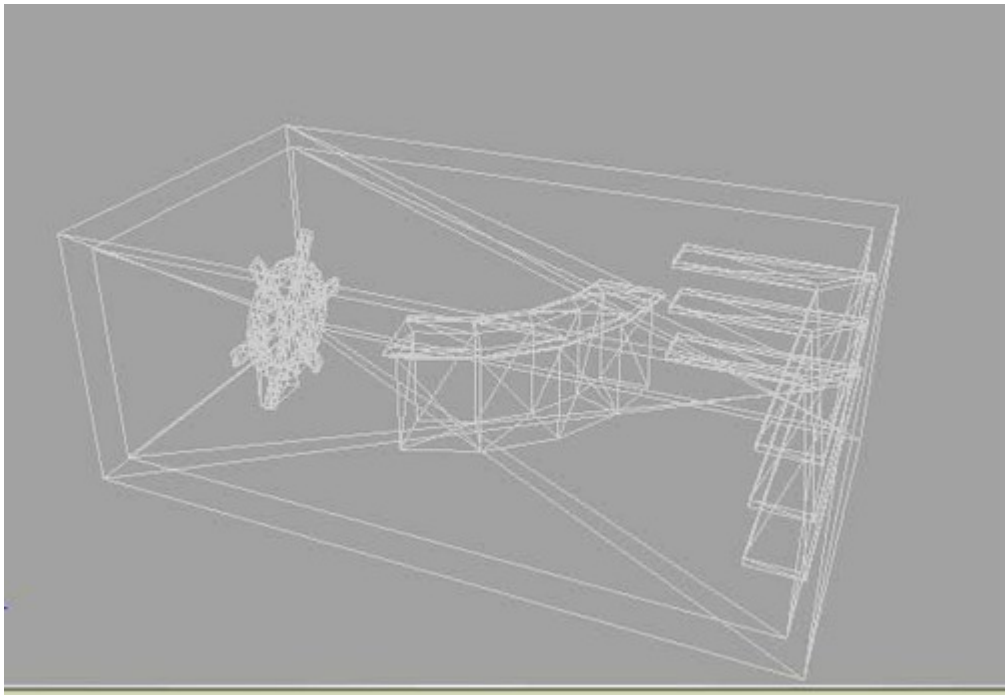
Anyway they are game specific and rather dull. I want to talk about a code issue.

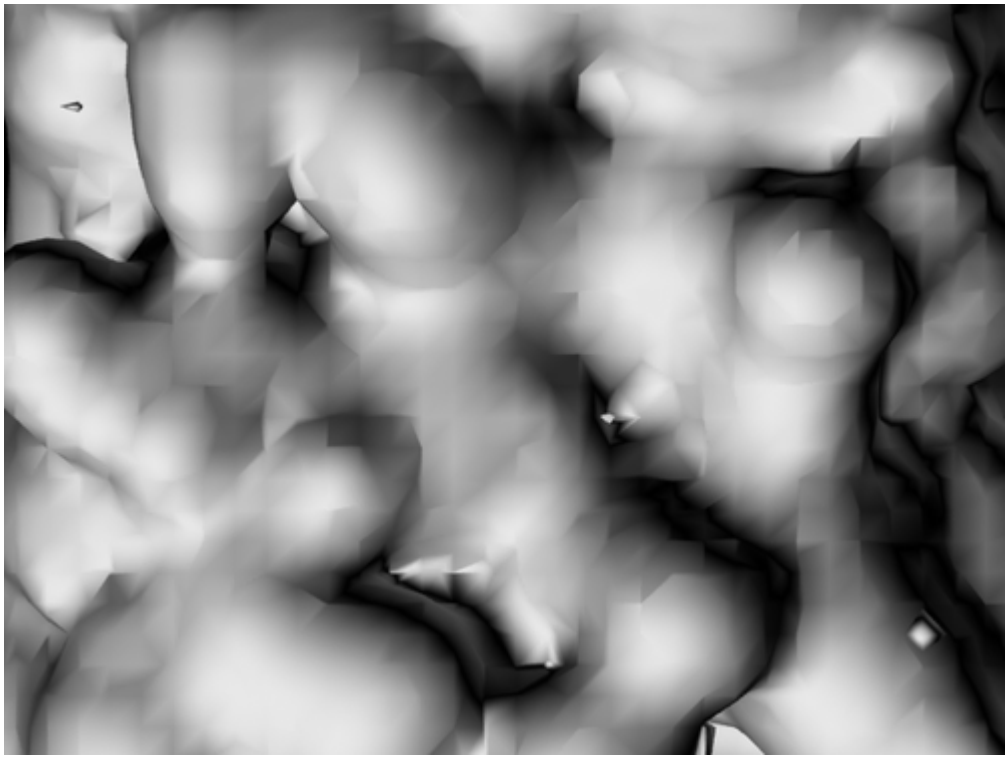
You want HOW MANY!

Beginners and hackers will always dynamically allocate verts and primitives. THIS IS BAD! I know its convenient but its wrong. Resources should be compiled and processed by the content pipeline or equivalent. This allows unified buffers, optimisation of vertex declarations and lots of other neat things which make the Tech Trolls happy, otherwise they club you with a big stick.
This applies to the UI because well I was looking at generating the menu based on X choices which means generating verts based on splits& blah blah blah.
Caveat: *Rules are made to be broken! Truly dynamic objects, like water simulations for instance may need to be altered. However try move choices like vert declarations and number verts to pack time. Understand why you are breaking the rule before breaking it. Remember over assigning a large pool on a loose fit is often better than a dynamic solution which fits better but restrains you more.*
Wow that was a long post.



Shop Blimp - Love Programmer Art

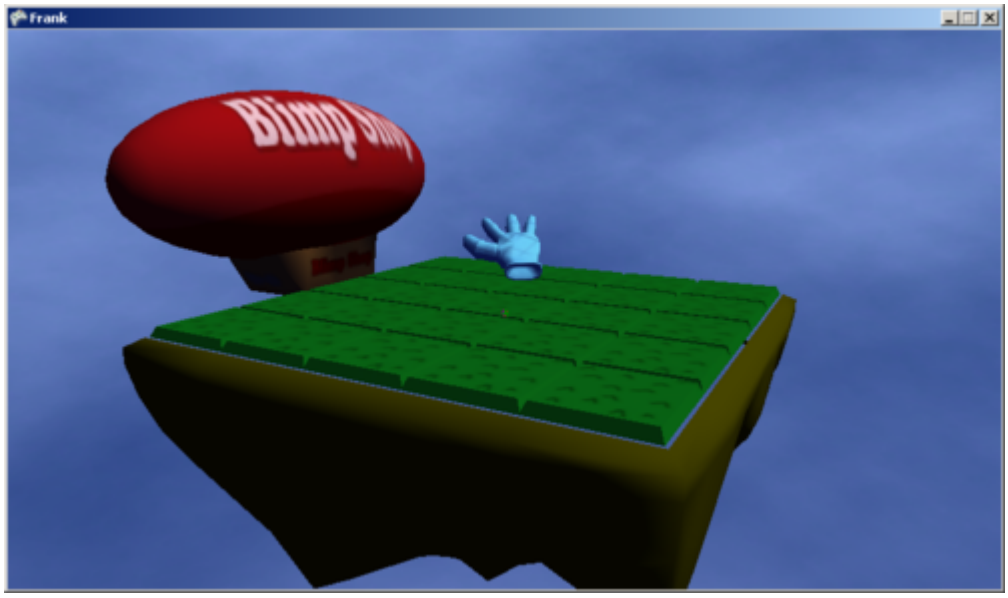

Shop Internals, rendered in wireframe

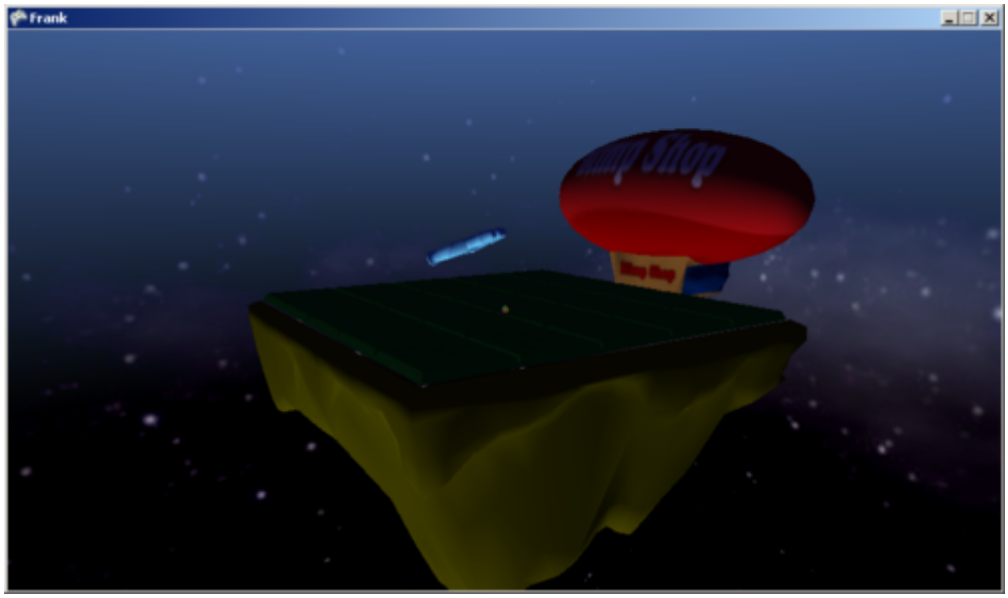


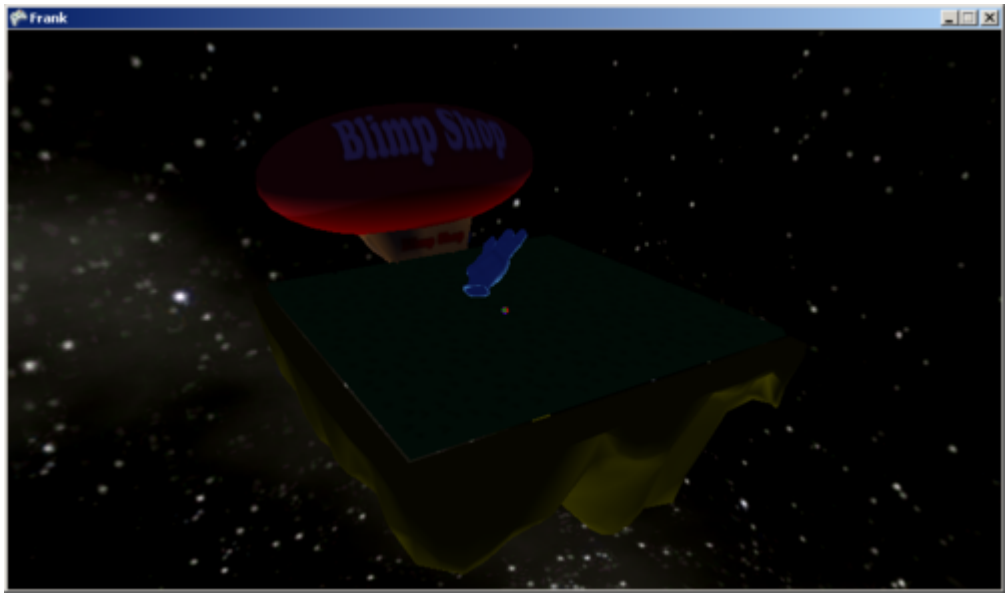
Ye gods captain, PROGRESS.
After the dismal run around the pointlessness I have actually made some progress last night and today.

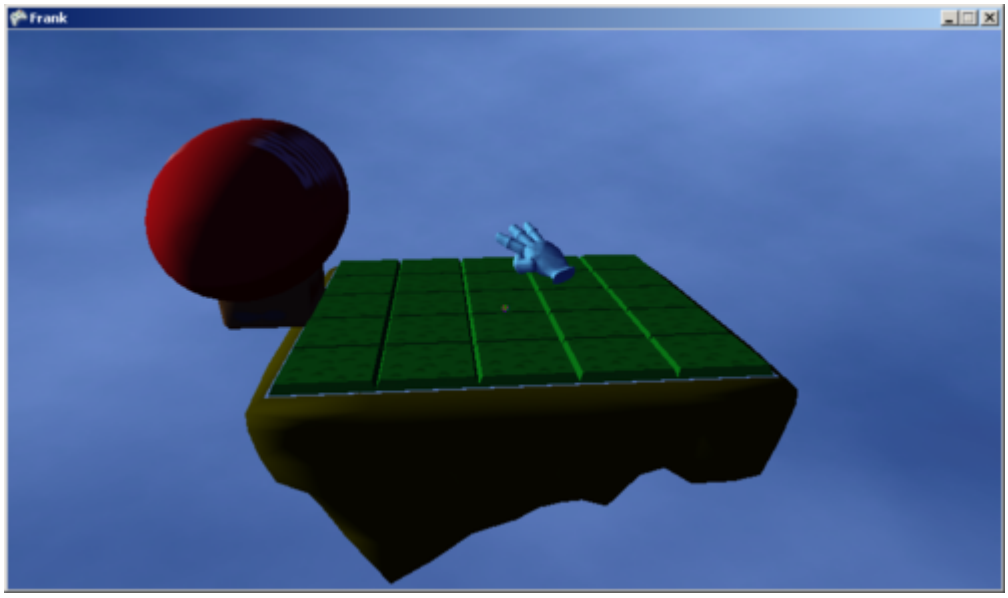
- Coupling Issue solved with Interface
- Island has day/night cycle
- Shop Blimp interior & exterior modelled
- Blimp added to world
- Add a reliable help to translate between tile and world space
- Started work on Sky Box

Coupling Issue
Okay so we dont want the model (island & prop data) coupled to the view (scene graph, animations, models). Well the solution was to give every node in the scene graph a user data element which had an interface.
The core functions being, Update & Pre-Render. Further functions and event handles can be added to the interface to handle animations and the like. The interface is the façade which prevents tight coupling. Neat and clean, kicking myself for not seeing this solution weeks ago.
Sky Box
My reason for posting, indirectly. I waster the last hour or so trying to generate normal & height maps for clouds. Reading up on things I already knew how todo if I stopped and thought for a second. The moment I saw my folly I stopped to write this to break me out of it.
I will now grab some food then hopefully not be stupid.


Frank
Blimp Shop


Frank


Frank
Blimp Shop



Day & Night Cycle
Woot! Not only did I get a pretty awesome Sky-Sphere in place but I also managed to knock up a day/night cycle. I also cleaned up the shaders and have global lighting in place.
Found an awesome tut, and manage to knock the shader up from that.
Next thing is to look at shadows, but dont you think that is preeety. :P

Lights Camera & Action

Well lighting & shadows are a complex bag of trickery and maths.
Basic Lighting is simple. Take a book outside into the sun now rotate the bugger around and throw it. You will notice the amount of light on a surface can be measure as an angle between the light direction (from the sun) and the facing direction of the surface. A very cheap and easy calculation we love, called **dot product**.
Now shadows are more complex. Light as a physics based thing in reality is a crazy thing which drove Einstein insane. For instance do you know light is affected by gravity so it actually bends around objects forming a very distinctive halo effect. So not only is it a ray cast but its a particle/wave which is affected by gravity fields. They also split up and bounce all over the place.
Now in games we fake it, cause those calculations are really expensive, and most times we care more about whether a bullet hit a guys head.

Bake It

A cheap and simply trick is to back the shadows. This works well if the lights & geometry dont move much. Even in AAA games with more advanced systems, baked light maps do a lot of the heavy lifting.
To bake it we basically do the calculation before the game runs. The simplest form is a flat planar map, some bake a projection map and some cases even bake shadow volumes. We can animate the baked textures and get some awesome effects.

Fake It

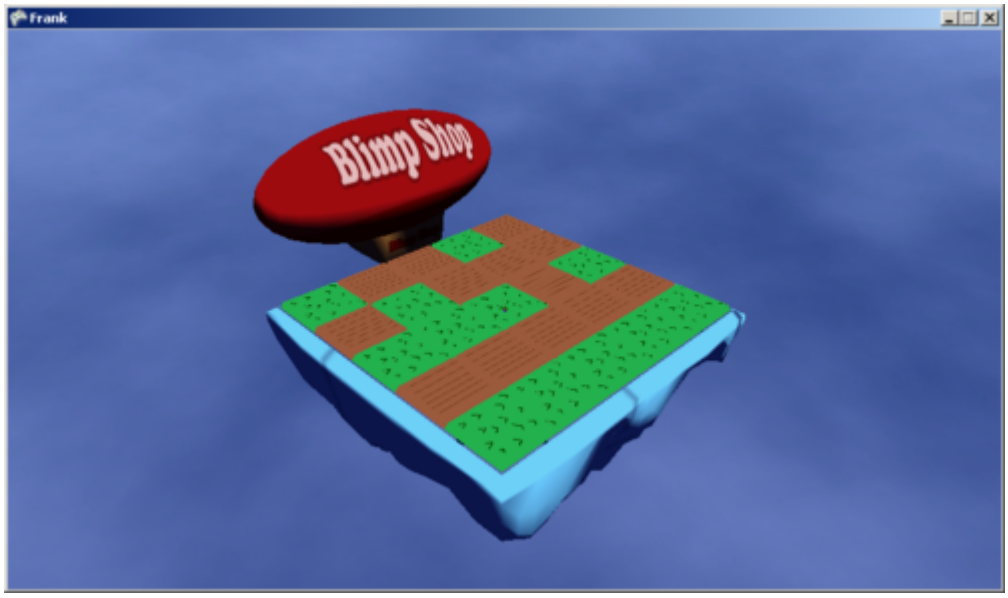
Okay so we have this bad guy, race car or some other object in the world. Well we can forgive a lot with our suspension of disbelief. What if we glued a blob shadow to the feet of bad guys or race car.
You would be amazed how many games use this system to optimise thing and it looks great.

Flatten It

Well if we simplify light and ignore the bending, bouncing, and craziness of it, then we can treat it as a straight line from the light. Now we like planes, planes are really nice simple flat things.
Okay so now we have this thing called the projection matrix which is part of the matrix of projecting a 3D object into a 2D image. Well if we use the same trick we can clone an object, then squish it into a projection and render it all black. We have a shadow, Peter Pan Style.

Deferred Volume

Okay well know in reality we need to drop shadows onto complex geometries. Now we also have this wonderful shader pipeline. Before we render the scene we render the entire world from the perspective of the light (as we would the camera). We then grab the depth buffer and save it.
This needs to be done once for each light.
Now this makes an image of how far the beam of light travels. Which we pass in to the next render pass, along with light matrix. Now if we take a point in space transform it by the Model-View-Projection as normal. Also Transform is by Model-Light-Projection. Then we compare the MLP read-out against the baked texture we can tell if the light reaches that point or not.
Another way of doing Volume Shadows is to use Stencil Buffers, this is the traditional way to do things and how most shadow volumes have been done in the last 10 years.



Shadows and Reworks

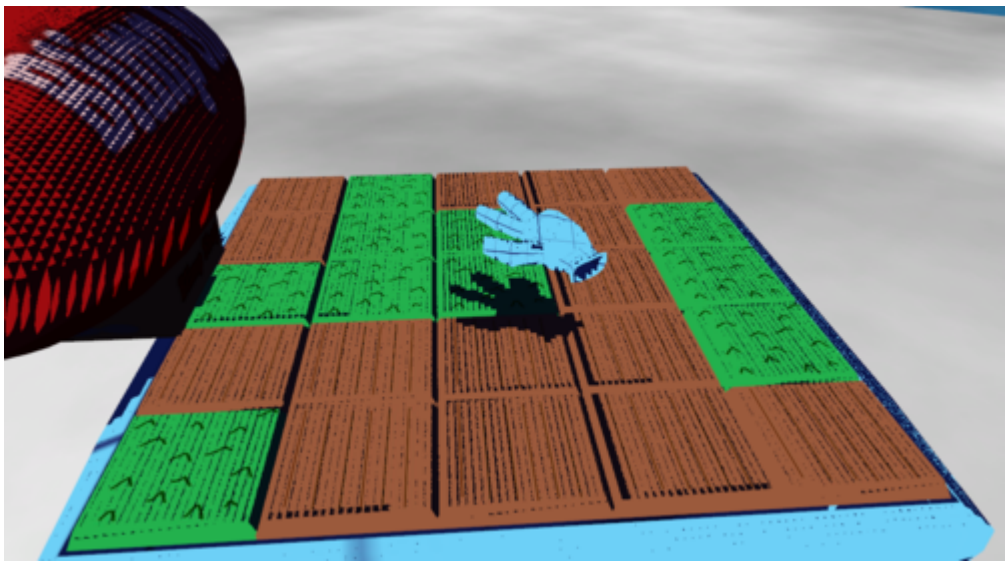
So after a lot of thought and planning yesterday I set about implementing soft shadows using shaders tonight.
Sadly this meant I needed to re-haul the entire scene manager and packer to support texture swapping, and effect management. I also had to write a big ass shader to do several things, and optimise.
Thats done, now all I need to do is add the Shadow Render pass and it all should work.
Im going to miss the last bus home, so I gotta run.

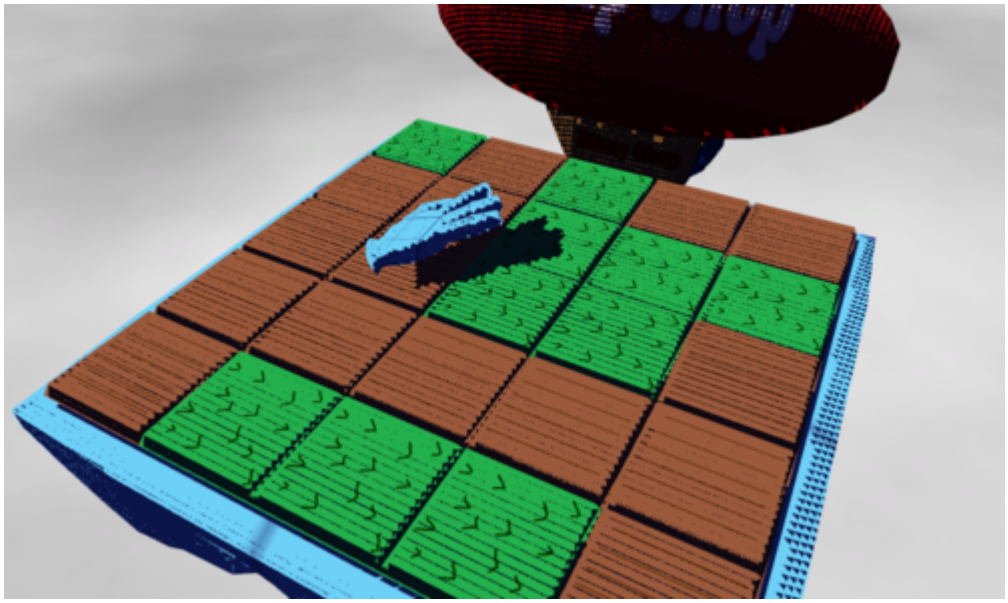
Full Screen shadows almost done. Had to run. Just caught last bus home. Some weird filming on the lot.
More tomorrow.

So climbing into bed last night I got a great idea.
Debug the Light ViewProjection matrix by using a camera with the same matrix. Well you know how I managed to knock together that skybox so quickly. Well I dug into it&. YIKES.
I managed to get the sky shader down from 64 instructions to 27 instructions, and remove 4 branches. Sadly the sun doesnt look great. On the bright side the Sky Sphere now respects its World Transform. Light amount now tied into horizon. Removed that horrible change line. Lost the tinting.
So this is what happened&

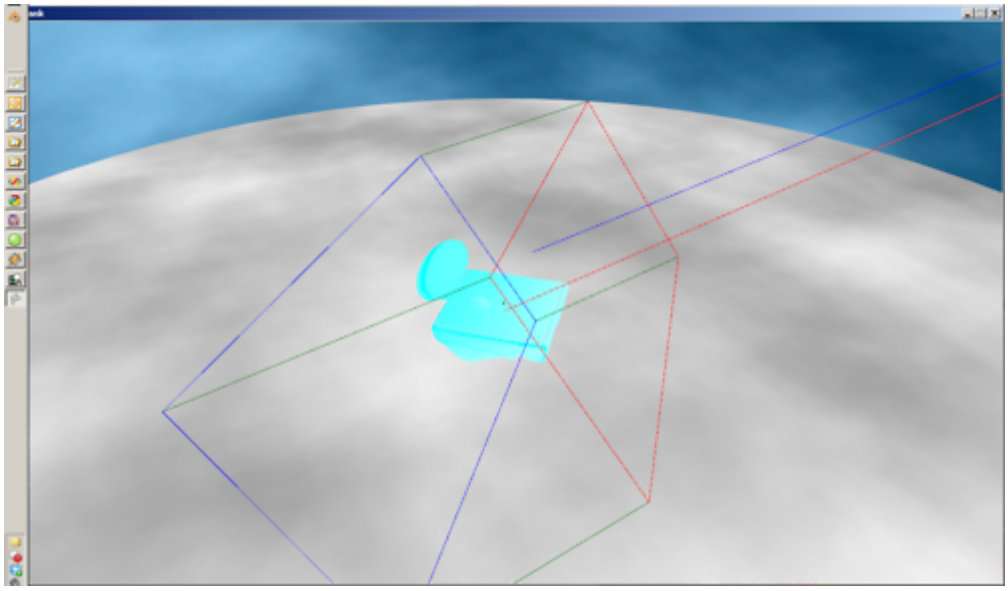
> Oh wait I can clean up those 2 lines quickly& later& Hmm this could be improved& later& wasnt I meant to be coding something?

Yeah so now after all that fiddling, Im now working on the shadows again.





Wooot!
Progress, the shadow is low quality due to PC card limits. I need to add some bias to avoid flicker, and smooth off the edges to get soft shadows.

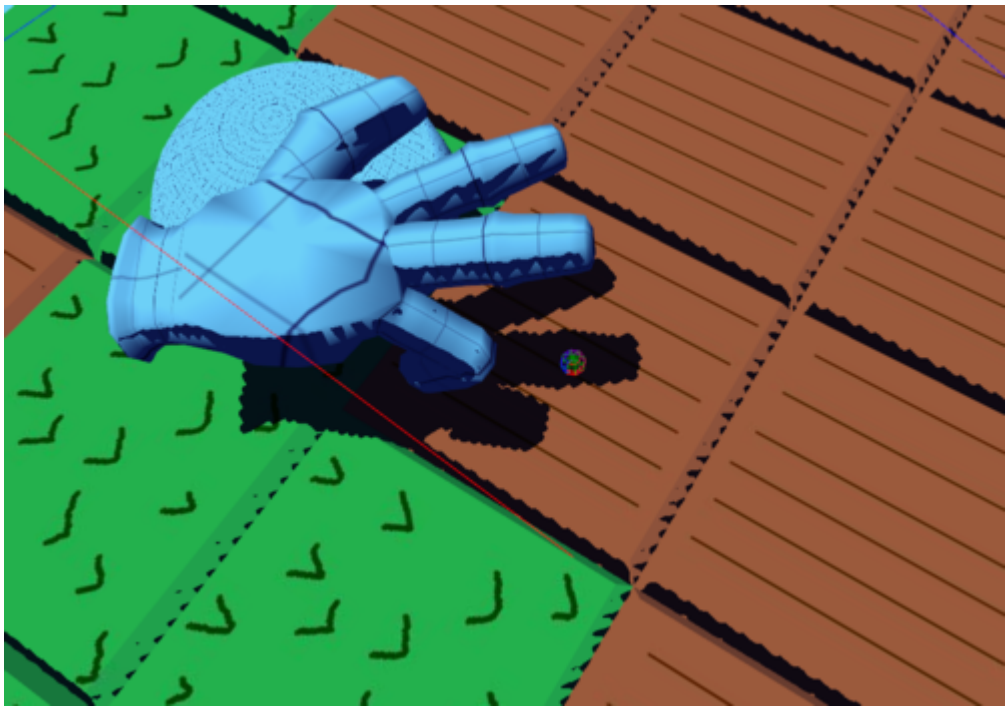


To Debug it& Draw it

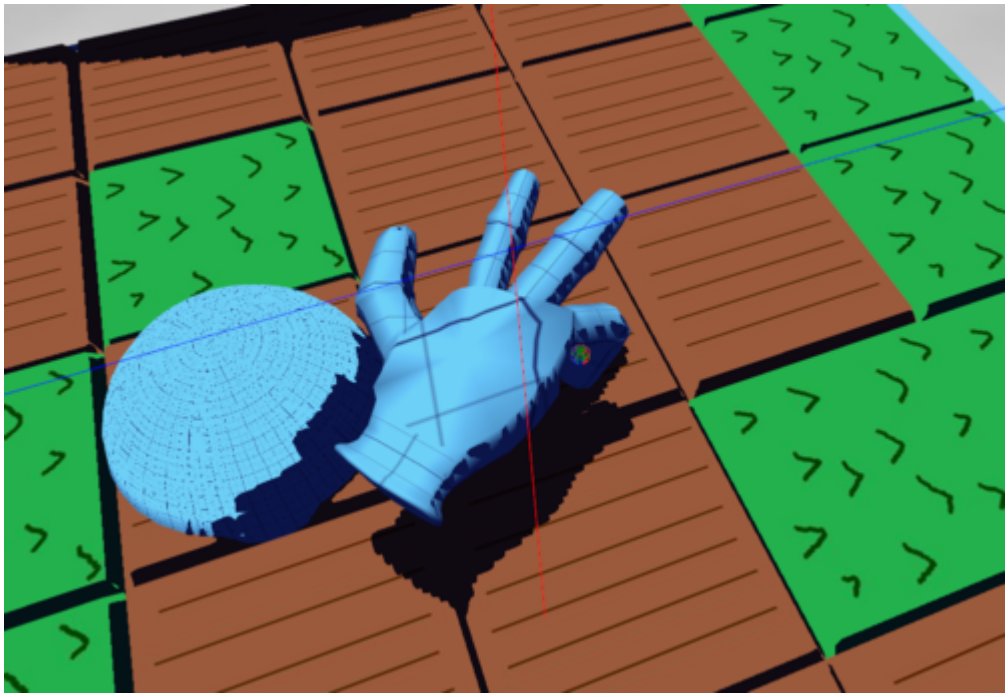
Many people will tell you this but whenever you have a problem try to visualise it. This can be applied to almost everything (except cases where visualisation is a constraint, think advanced maths and quantum). So above you see the debug frustum.
Well after fighting with all these various problems Im thinking to myself& I need to double check my Frustrum. Five minutes to write and **BAM** problem is instantly visible.
The problem I was debugging for 5 hours solved in 5 minutes.
Printf & Debug Draw rocks.

Problem: Shadows giving strange LightViewSpace results

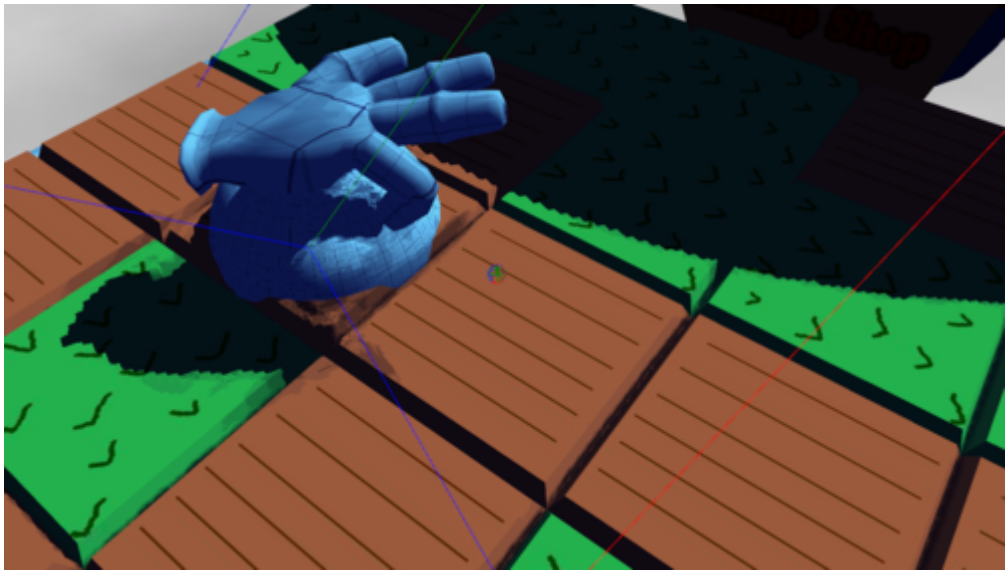
Well the problem manifested in many ways. I was debugging shaders and tweaking render targets. Not to mention twiddling the depth bias so often that it was indecent.
The problem was two fold. Firstly the sun view & position didnt match up. The result of a quick hack to get the sun in the right place, hacks bad. The second issue was todo with the scene bounds.
The view matrix is built with a look-at call. Well the key is that the centre of the bounds must be equal to the look-at target. Why is this? Well draw some random shapes on an A4 page. Now draw a circle which contains all those objects. Now measure the radius of the circle and draw a square at the centre of the page with double that length per side. You see the problem, unless the circle is at the centre of the page their will be bits of it outside of the box, which means they wont be shadowed.
So you compensate for offset and increase the radius. To do this instead of measuring the radius of the circle measure the distance from the centre of the page to the furthest point of the circle.
Before you ask we cant move the box from the centre of the page as that would change the angle the light was coming from.
Onwards and upwards.



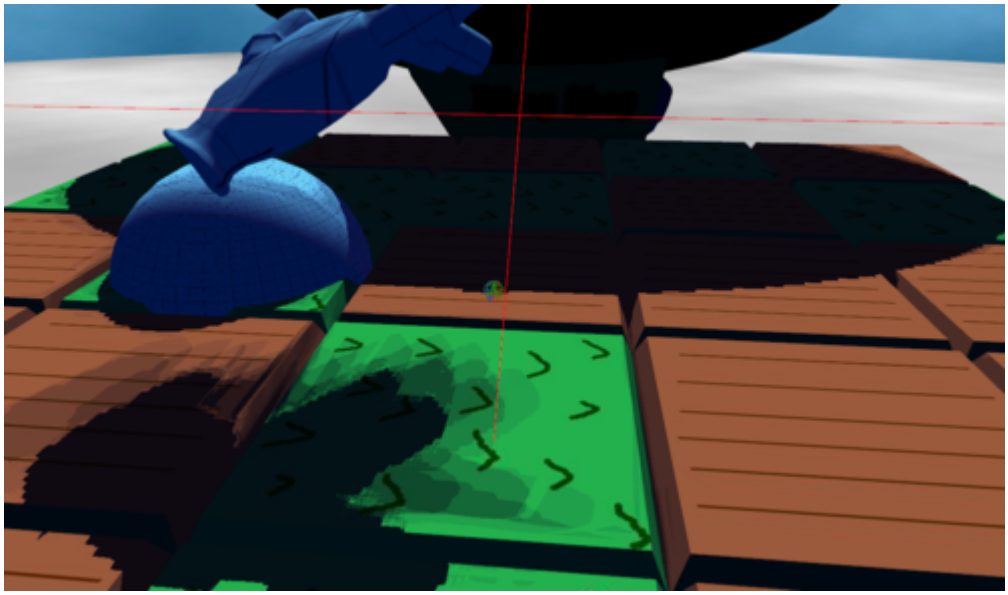
Rough Shadows



Rough Shadows



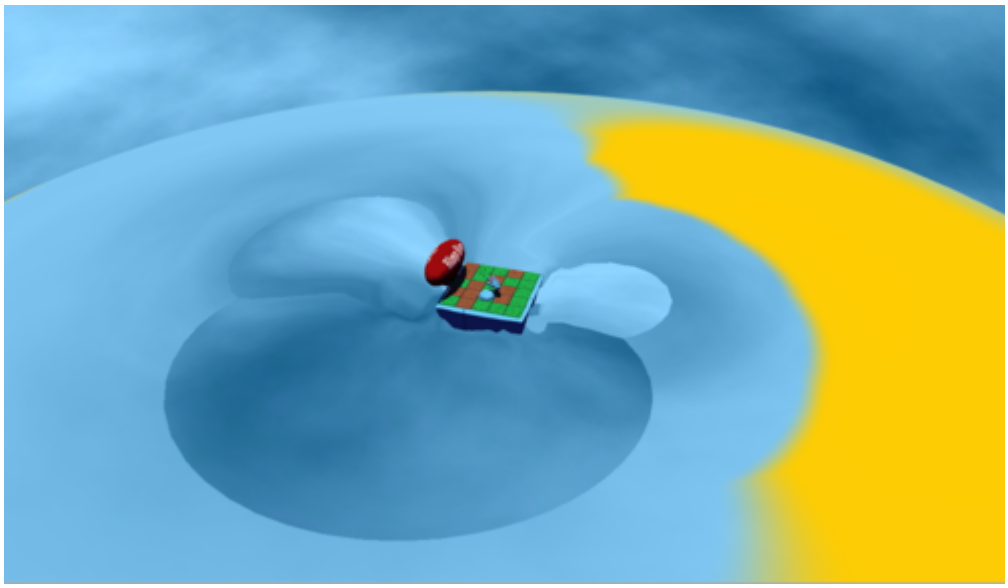
Blurred Shadows



Blurred Shadows

Shadows are EVIL!

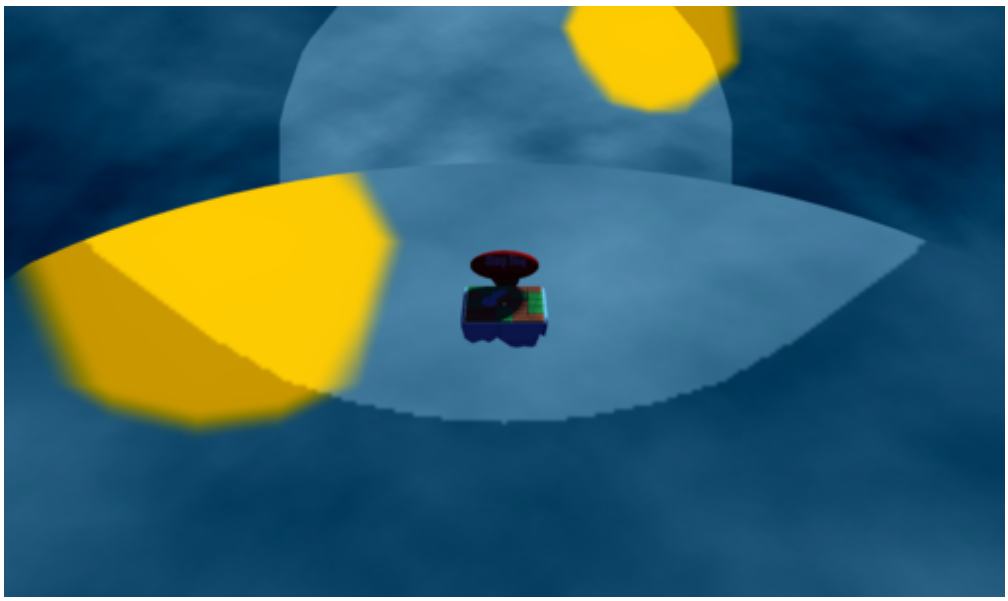
Okay above you see rough and blurred. The rough pass uses a single sample against the depth buffer. Those bloody pixels are a result of the resolution of the shadow map.
The blurred insanity is a shadow method called PCSS, based on the White Paper, Integrating Realistic Soft Shadows into Your Game Engine by Kevin Myers, Randima (Randy) Fernando, & Louis Bavoil. Sadly some of there stuff uses Shader Model 4 so I did my best approximation.
Not happy with either method& pulling my hair out.



Water

So for those of you curious Im throwing down some water rendering magic. Not to mention real-time environment cube mapping. Behold the result.

So its broken and the ocean needs some motion (*pun fully intended*). I also may be slightly sleep deprived& but onwards and upwards.



Cube Maps Ahoy

So there is a slight glitch with the reflection but ALMOST there.