V.E.D.I.C

Virtual Environment for Developing Interactive Code

Developers

William Funk
Jorge Rodriguez
James Vinson

Sponsor

Dr. Richard Leinecker

Project Narrative

Overview Of The System

Users of the VEDIC system will typically be programmers working with C#. These software developers would range anywhere from the novice, learning to program in an object-oriented language, to the experienced developer working on a complex project that involves numerous files and classes.

VEDIC will have a total of four modes of operation: the viewing of a database (SQL), the creation or manipulation of a database (SQL), the viewing of a program with its multiple files (C#), and the creation or manipulation of a program (C#).

In the viewing of a database mode, the user will see a multitude of different shapes of varying colors, some interconnected, some stand-alone. These will consist of SQL tables with distinct sections representing columns within these tables. Both tables and columns will be labeled and their colors will be specific to the type of variable they contain (integer, float, char, etc.). Colored pipes or wires will connect these tables where relationships exist (joinings, shared variables, etc.). The number of rows in the table will either exist as a number adjacent to the table's title, or in the relative height of the object in scale with the other table shapes. User will be able to rotate the scene or a specific table, expand a table for better view of its individual columns and possible list of row contents, zoom in and out, explode scene (increase distances between objects), implode the scene (decrease distances between objects), and move through the scene in a flying-style of motion.

The creation and/or manipulation of databases mode will have the user seated at a virtual work-bench. On a wall directly in front of the desk/bench, and within arms-reach, will be a series of shelves with cabinets, bins, drawers, or boxes labeled with the corresponding object(s) they contain, with the image of the representative shape(s) on the front (ie. a cyan-colored, flat, rectangular cube representing a column that contains an integer type). On the desk/work-bench in front of the user will be the complex object they are building or manipulating (much like a 3-dimensional circuit diagram). This object/SQL-table will hover above the work-bench and can be connected using specifically colored wires/pipes to other tables and their columns. The user can change between view and edit modes at will. This mode will comply with the standard C.R.U.D. array of database operations: database tables can be Created, data can be Read from rows by columns (more easily in the view mode). Tables, columns, and rows can be Updated while any of those can be Deleted, as well. The system will implement one query at a time as the user submits them.

Third VEDIC mode is the viewing of a C# project. This mode will contain colored shapes and connections similar to the viewing of the database mode. The difference will be the number of different shapes and colors as a programming project has a more diverse set of components than that of an SQL database. This view will share many similarities with a UML diagram, both structural and behavioral, with a third dimension to better illustrate the complexity and interaction that the objects possess.

The fourth and final VEDIC mode of operation is the creation and/or manipulation of C# files and classes within a project. Similar to the create/edit mode for databases, there will be many more shapes, colors, and connections to add into the 3D circuit floating above the work-bench. These classes, their functions and variables, can be transpiled into an export format when the user elects to do so, and converted to C# files in a project folder for later compiling by the user outside of the VEDIC system.

VEDIC software will ultimately be free to use under the MIT license. It is this group's belief that freely offering the software will promote its use, and bolster its circulation in a shorter amount of time.

Functionality

- Represent a SQL database and allow CRUD application to said interface in a visually intuitive way.
- Identify and display connections between SQL tables and their contents in a way not possible with two dimensions.
- Represent a series of C# files/classes/objects/etc. in a 3D virtual reality context that better portrays an object-oriented language.
- Represent a 3D programming environment and allow the user to create/manipulate the environment in the development of their own code/systems.
- Assemble function declarations using a jigsaw puzzle of components.
- Identify dependencies to a class in a visual context (interfaces and abstract classes).

Motivations

- *William Funk*

It was one of my earliest childhood memories where I had first been introduced to virtual reality 25 years ago. In the carnival games section at Busch Gardens, there was a featured attraction. It had the size of a small pool, a headset that threatened to topple me over from the sheer weight of it, and a handheld blaster reminiscent of the science fiction movies of that era. I can still remember that 3D world of polygons, low-resolution pterodactyls swooping in, and my cube projectiles spitting forth at a whopping 40 frames per second. For a moment, as a child, I had stepped into the future; an exciting future with endless possibilities. Now two-and-a-half decades later, I've finally caught up to that future. I can create the reality around me with the code from my fingertips. What's left? Someone or some group to make this technology more relevant in the here and now. *This* is why VEDIC is important. This is *our* contribution to that dream.

- *Jorge Rodriguez*

My initial motivation to pursuing this senior design project was to create a playground where people can come in and interact with a system that we provide on a social level. This eventually molded itself, with the help of my colleagues, into VEDIC. I love the idea of working with an engine like Unity, the inclusion of VR equipment such as the Oculus, and other tech to make something extraordinary. I have always been drawn to videogames, and working in Unity for this project was an obvious candidate for projecting that lifelong stimulation into a profession. Of course, this goes beyond just working in Unity, because the project is trying to utilize the potential of Unity as a "game-engine" to create a platform for developing code.

- *James Vinson*

Blah, Blah, Blah. Make something cool. My name James!

Broader Impacts

Big Data is a phenomenon that grows with each technological advance in storage space, processor speeds, broadband width, and mathematical algorithm. The caveat is how to view and interpret that large data set in a meaningful way. Many have tried to use 2D node graphs with lines criss-crossing in a convoluted web that takes considerable time to untangle.

VEDIC provides an intuitive way to more accurately represent that data. Skyscraper-sized SQL tables could tower over a one-story house-sized counterpart to display

volumetric relationships with no end to scalability. It's accuracy would rival any SQL database GUI (ie. MySQL Workbench), but with the added benefit of a flexible appearance that the user could customize for better individual satisfaction. Are there too many connections to a single table? Would it be better to create two tables in order to shunt some of that traffic away from a single location? With VEDIC, a user could see the vast network of dependencies snaking their way from one table to the next like power lines in a dense urban area. In addition to the viewing of all that data, various queries can also be better visualized as inserting something into a table is *actually* inserted into the table by way of the user's hand.

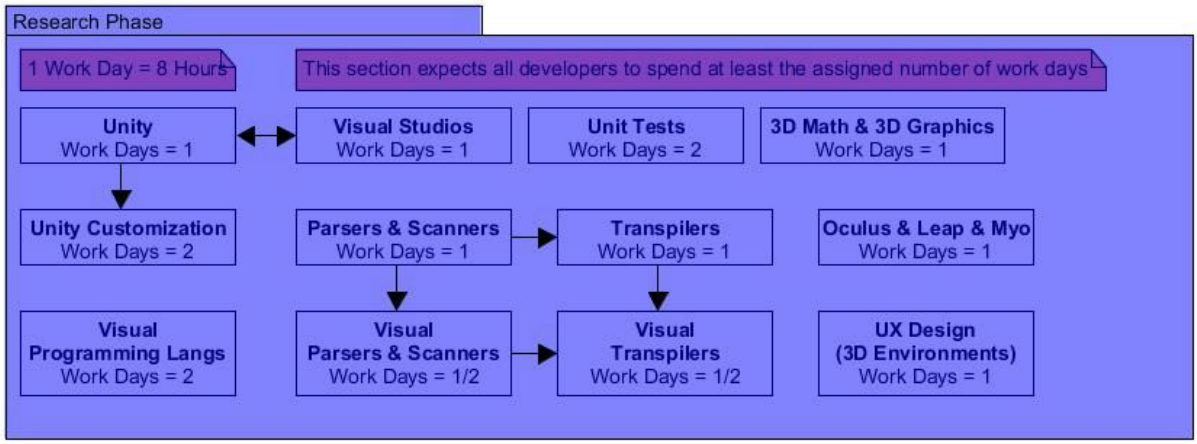
Object-Oriented Programming by its nature is three-dimensional. The world has been forced to code in only two up to this point, while novice programmers often find it difficult to transition from procedural languages because of a lack in visual representation to what effectively amounts to real-world objects and their attributes.

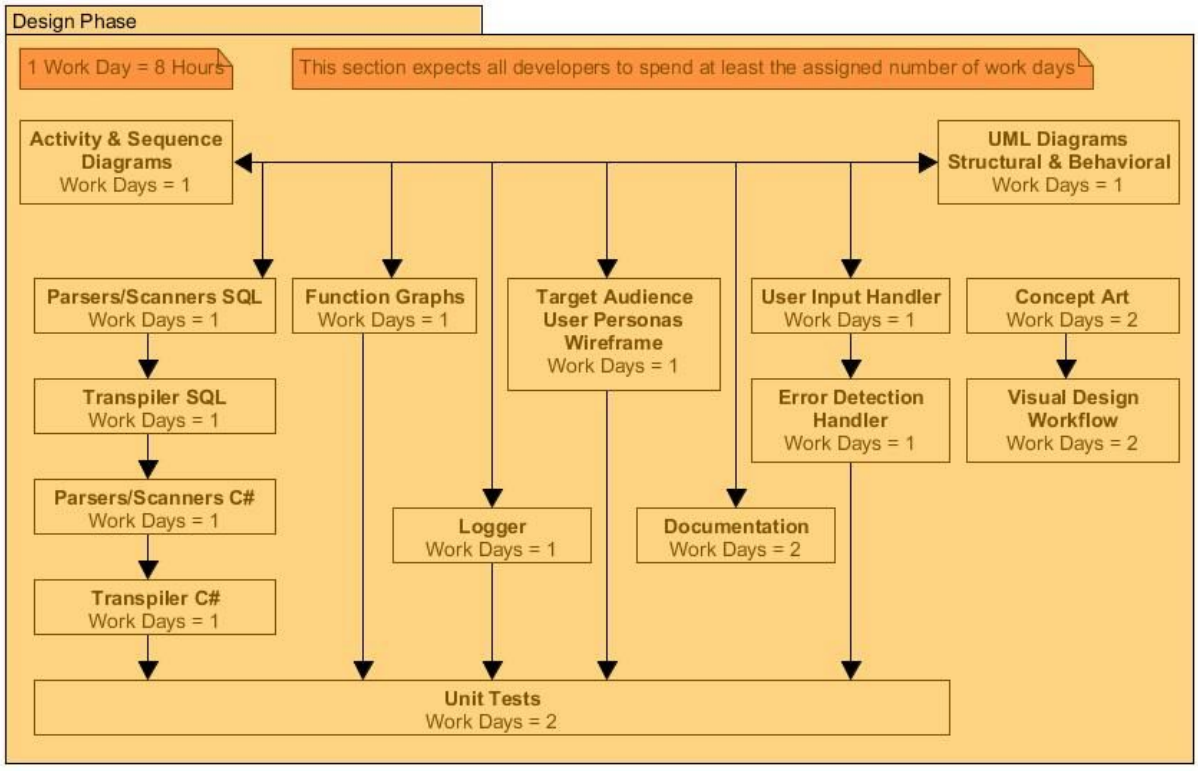
No longer will a computer science student be forced to learn using the archaic system we now rely on. They can completely immerse themselves in the experience of programming by taking hold of the objects themselves, actually insert variables into them, and piece together functions from a comprehendible array of nodes. This will increase the speed at which the programmer codes, make for more robust programs, and give debugging a face.

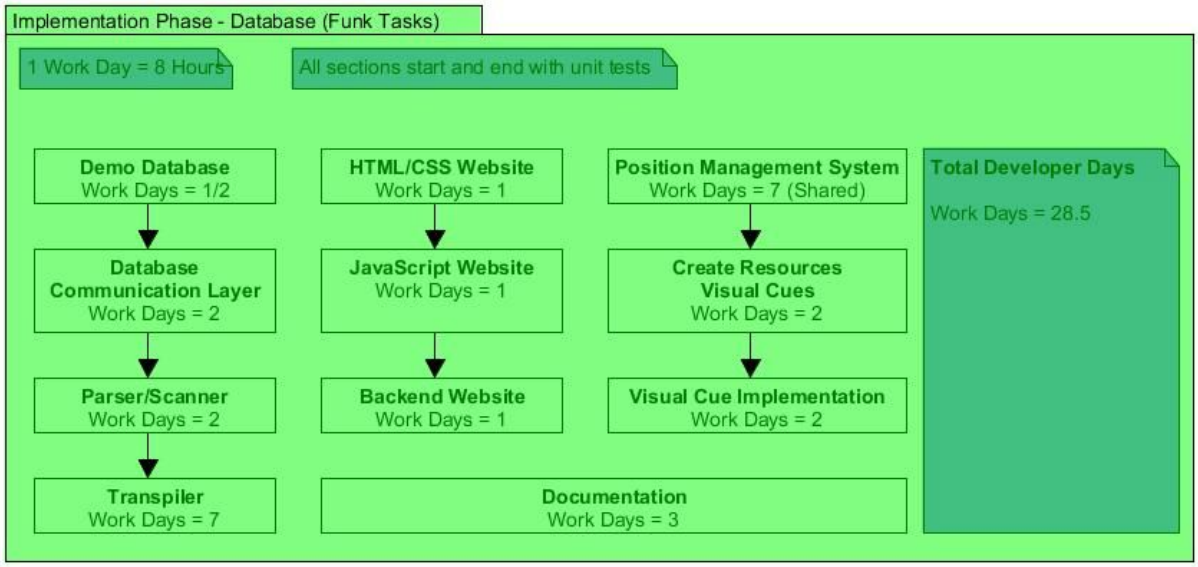
Requirements

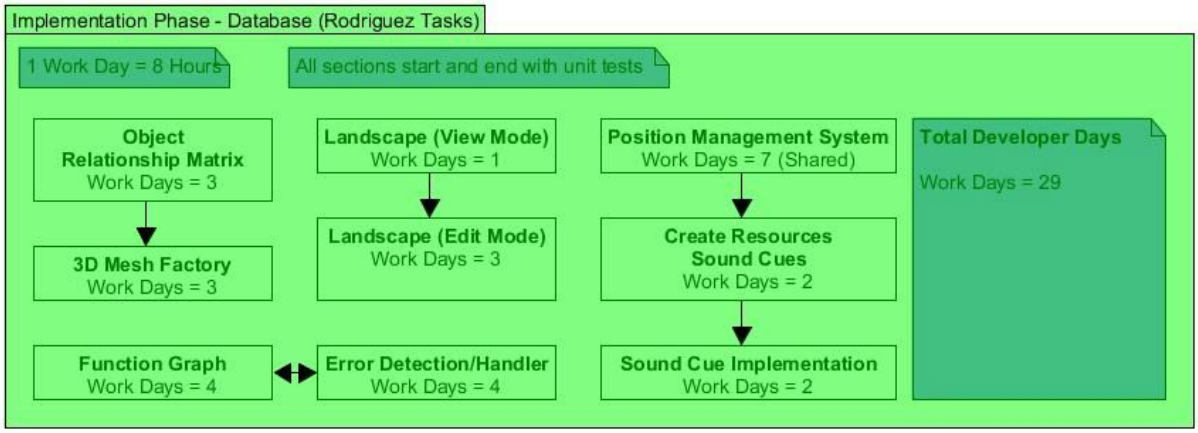
Design Plan

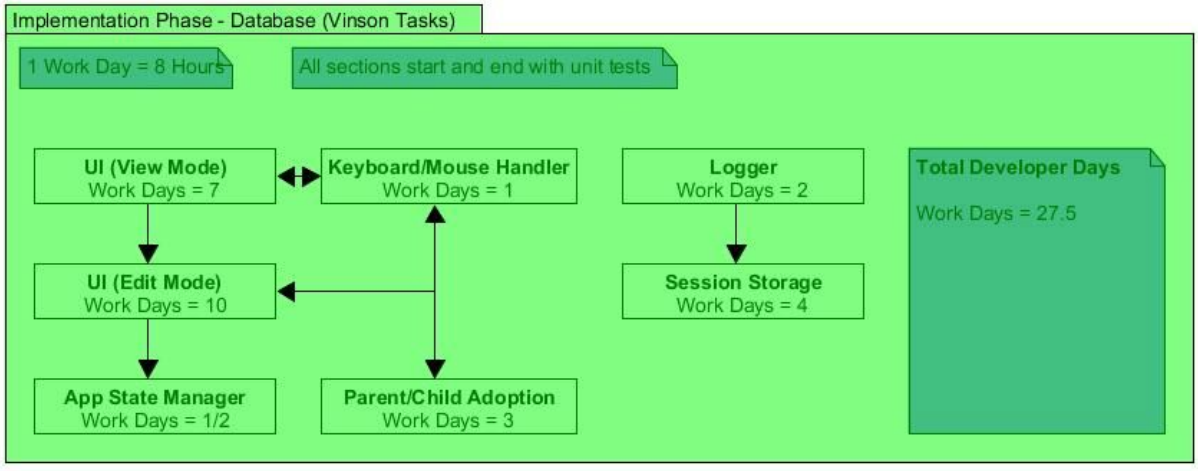
Task Diagrams

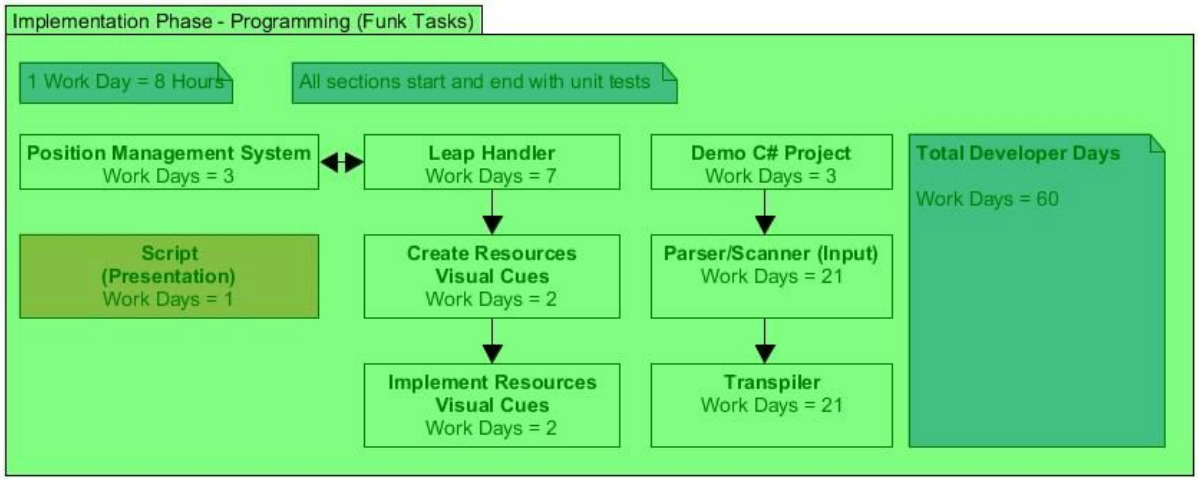


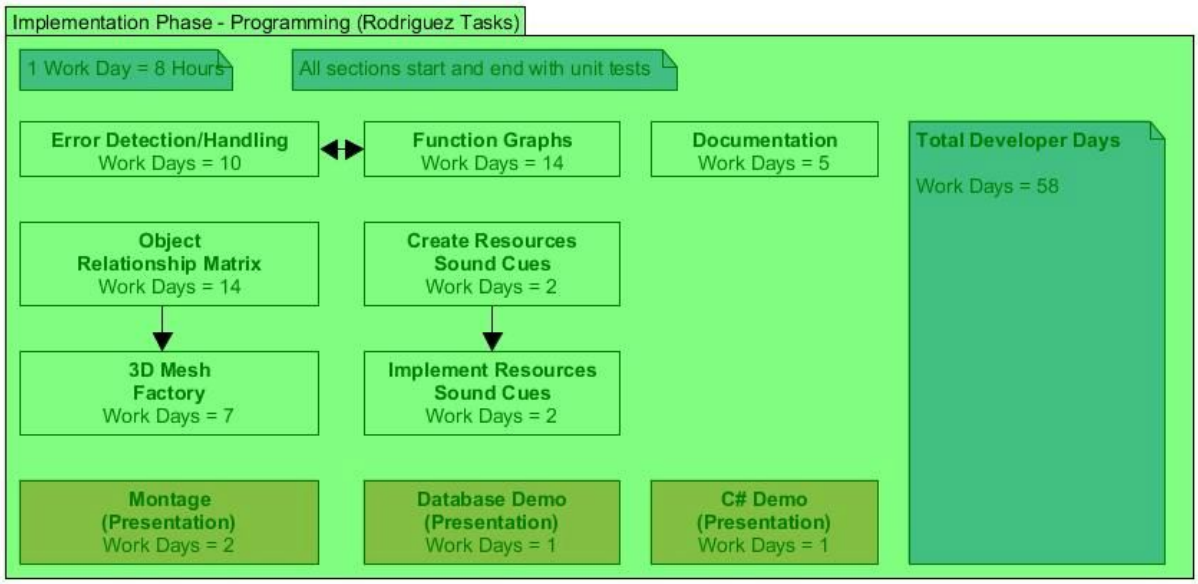


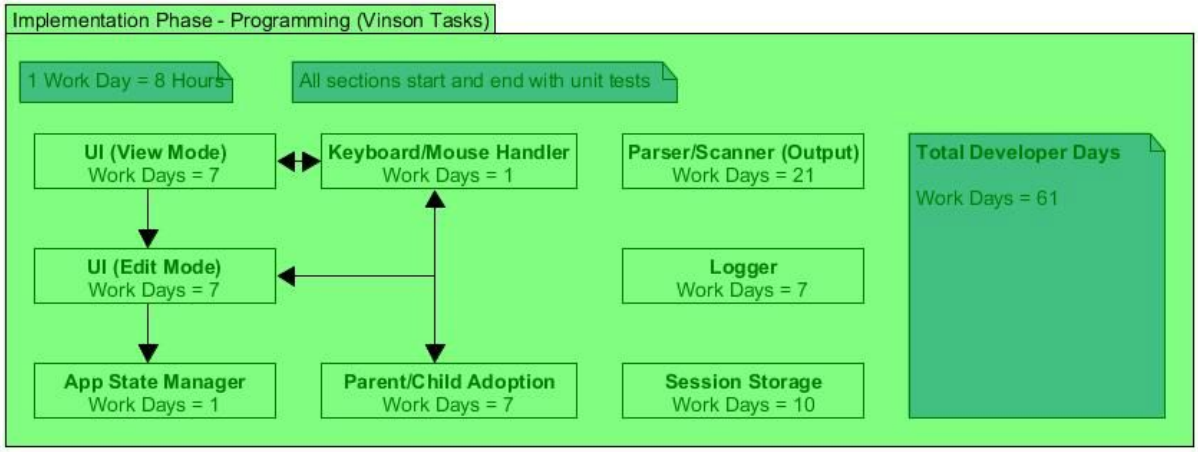

Implementation Phase - Database (Funk Tasks)
1 Work Day = 8 Hours
All sections start and end with unit tests
Demo Database
Work Days = 1/2
Database Communication Layer
Work Days = 2
Parser/Scanner
Work Days = 2
Transpiler
Work Days = 7
HTML/CSS Website
Work Days = 1
JavaScript Website
Work Days = 1
Backend Website
Work Days = 1
Documentation
Work Days = 3
Position Management System
Work Days = 7 (Shared)
Create Resources Visual Cues
Work Days = 2
Visual Cue Implementation
Work Days = 2
Total Developer Days
Work Days = 28.5


Implementation Phase - Database (Rodriguez Tasks)
1 Work Day = 8 Hours
All sections start and end with unit tests
Object Relationship Matrix
Work Days = 3
3D Mesh Factory
Work Days = 3
Function Graph
Work Days = 4
Landscape (View Mode)
Work Days = 1
Landscape (Edit Mode)
Work Days = 3
Error Detection/Handler
Work Days = 4
Position Management System
Work Days = 7 (Shared)
Create Resources Sound Cues
Work Days = 2
Sound Cue Implementation
Work Days = 2
Total Developer Days
Work Days = 29


Implementation Phase - Database (Vinson Tasks)
1 Work Day = 8 Hours
All sections start and end with unit tests
UI (View Mode)
Work Days = 7
UI (Edit Mode)
Work Days = 10
App State Manager
Work Days = 1/2
Keyboard/Mouse Handler
Work Days = 1
Parent/Child Adoption
Work Days = 3
Logger
Work Days = 2
Session Storage
Work Days = 4
Total Developer Days
Work Days = 27.5


Implementation Phase - Programming (Funk Tasks)
1 Work Day = 8 Hours
All sections start and end with unit tests
Position Management System
Work Days = 3
Leap Handler
Work Days = 7
Demo C# Project
Work Days = 3
Total Developer Days
Work Days = 60
Script
(Presentation)
Work Days = 1
Create Resources
Visual Cues
Work Days = 2
Parser/Scanner (Input)
Work Days = 21
Implement Resources
Visual Cues
Work Days = 2
Transpiler
Work Days = 21


Implementation Phase - Programming (Rodriguez Tasks)
1 Work Day = 8 Hours
All sections start and end with unit tests
Error Detection/Handling
Work Days = 10
Function Graphs
Work Days = 14
Documentation
Work Days = 5
Total Developer Days
Work Days = 58
Object
Relationship Matrix
Work Days = 14
Create Resources
Sound Cues
Work Days = 2
3D Mesh
Factory
Work Days = 7
Implement Resources
Sound Cues
Work Days = 2
Montage
(Presentation)
Work Days = 2
Database Demo
(Presentation)
Work Days = 1
C# Demo
(Presentation)
Work Days = 1


Implementation Phase - Programming (Vinson Tasks)
1 Work Day = 8 Hours
All sections start and end with unit tests
UI (View Mode)
Work Days = 7
Keyboard/Mouse Handler
Work Days = 1
Parser/Scanner (Output)
Work Days = 21
Total Developer Days
Work Days = 61
UI (Edit Mode)
Work Days = 7
Logger
Work Days = 7
App State Manager
Work Days = 1
Parent/Child Adoption
Work Days = 7
Session Storage
Work Days = 10

Illustration

Cost Projections

Assets Acquired

-	Oculus Development Kit 2.0	---	$550.00
-	Leap Motion Controller	---	$30.00
-	PS4 Controller	---	$60.00
-	XBOX One Controller	---	$60.00
			$700.00

Projected Acquisitions

-	2nd Leap Motion Controller	---	$30.00
-	Code School Tutorials	---	$50.00
-	Leap Motion Keyboard	---	$100.00
-	Myo Band Controller	---	$200.00
-	3D Assets	---	$100.00
			$480.00

Projected Total

-	All Assets	---	$1180.00

Milestones

Research Phase (March 13th, 2016)

- **Unity (All Developers)** - February 28th, 2016
  Learning the Unity environment through tutorials and the creation of demo projects.

- **Visual Studios (All Developers)** - February 28th, 2016
  Learning the Visual Studios IDE by using it alongside the above mentioned Unity projects.

- **Visual Programming Languages (All Developers)** - March 6th, 2016
  Reading the existing literature and research papers on the subject and looking at “current system” examples.

- **Unit Tests (All Developers)** - March 6th, 2016
  Learning how to program unit test in general C#, as well as how to do so specifically in Unity.

- **3D Math & 3D Graphics (All Developers)** - March 6th, 2016
  Re-learning the basics of math and graphics in three dimensions through reading academic literature.

- **Unity Customization (All Developers)** - March 6th, 2016
  Watch tutorial videos on how to customize the Unity edit environment, and follow along in demo projects.

- **Parsers & Scanners (All Developers)** - March 13th, 2016
  Re-learn how to program parsers and scanner through class notes, and additional reading on C# versions.

- **Transpilers (All Developers)** - March 13th, 2016
  Learn how transpilers are programmed, and research if SQL and/or C# versions already exist.

- **Visual Parsers & Scanners (All Developers)** - March 13th, 2016
  Reading the existing literature and research papers on the subject and looking at “current system” examples.

- **Visual Transpilers (All Developers)** - March 13th, 2016
  Reading the existing literature and research papers on the subject and looking at “current system” examples.

- **Oculus, Leap, & Myo APIs (All Developers)** - March 13th, 2016
  Read the official documentation on these systems’ API, and watch tutorials on current uses of them.

- **UX Design for 3D Environments (All Developers)** - March 13th, 2016
  Read existing literature, and watch tutorial videos on how to design intuitive user interfaces in 3D.

Design Phase (April 17th, 2016)

- **Target Audience, User Personas, Wireframe (All Developers)** - March 13th, 2016
  Layout the look, feel, and expected use of the VEDIC home site.

- **Activity & Sequence Diagrams (All Developers)** - March 20th, 2016
  Expected activities of users, and the order of events they activate/encounter, followed by expected outcomes.
.

- **UML Diagrams - Structural & Behavioral (All Developers)** - March 20th, 2016
  Programmatic outline of classes, attributes, and methods of the program every step of the way (all states).

- **Parser/Scanner SQL (All Developers)** - March 27th, 2016
  Outline the flow and functions involved in the scanning and then parsing chain of events to/from an json string.

- **Transpiler SQL (All Developers)** - March 27th, 2016
  Outline the flow and functions involved in the transpiling of parsed tokens into quantities of prefab object types.

- **Parser/Scanner C# (All Developers)** - March 27th, 2016
  Outline the flow and functions involved in the scanning and then parsing chain of events to/from a C# file..

- **Transpiler C# (All Developers)** - March 27th, 2016
  Outline the flow and functions involved in the transpiling of parsed tokens into quantities of prefab object types.

- **Function Graphs (All Developers)** - April 3rd, 2016
  Identify the modular components in a function declaration, design each in C#, and outline their relationships.

- **Logger (All Developers)** - April 3rd, 2016
  Outline specific components of a log, and where the data is stored. Outline the infrastructure for those logs.

- **Concept Art (All Developers)** - April 3rd, 2016
  Commission a series of high-end, graphic designs of the four modes of operation in stylized fashion.

- **Visual Design & Workflow (All Developers)** - April 10th, 2016
  Create an organized matrix of visual shapes/colors that associate with specific classes/tables/variables/etc.

- **User Input Handler (All Developers)** - April 10th, 2016
  Pre-define the keyboard keys, mouse events, and hand gestures to their respective in-program responses.

- **Error Detection Handler (All Developers)** - April 10th, 2016
  List out the predicted scenarios a user could go wrong, and prevent those actions or design the responses.

- **Unit Tests (All Developers)** - April 17th, 2016
  Outline the template pattern of a unit test, and layout the specific instantiations of all other tasks.

- **Documentation - Forms (All Developers)** - April 17th, 2016

Outline the template of the forms used in our future documentation. An online form could handle all tasks.

Implementation Phase - Database (June 12th, 2016)

- **HTML/CSS Website (William Funk)** - March 13th, 2016
  Build the layout and style of VEDIC informational landing site.

- **JavaScript Website (William Funk)** - March 13th, 2016
  Implement the front-end functionality of the VEDIC informational landing site.

- **Backend Website (William Funk)** - March 13th, 2016
  Create the php backend for the VEDIC informational landing site, and create the rudimentary SQL database.

- **Demo Database (William Funk)** - March 13th, 2016
  Setup and populate an SQL database for use in the VEDIC database modes during testing and presentation.

- **Database Communication Layer (William Funk)** - April 3rd, 2016
  Create the php/.NET backend layer for the demo database for testing and presentation.

- **Parser/Scanner (William Funk)** - May 1st, 2016
  Implement the SQL scanner and parser used to tokenize the json string incoming, and the 3D objects outgoing.

- **Transpiler (William Funk)** - May 1st, 2016
  Implement the SQL transpiler used to turn tokenized SQL data into 3D prefab object types.

- **Position Management System (William Funk & Jorge Rodriguez)** - May 8th, 2016
  Create the functionality that manages how objects move through the 3D space relative to a user input.

- **Create Visual Cue Resources (William Funk)** - May 15th, 2016
  Develop animated visuals to illustrate the various events taking place within VEDIC's 3D environment.

- **Implement Visual Cue Resources (William Funk)** - May 29th, 2016
  Place the visual cue resources in their respective event handlers.

- **Documentation (William Funk)** - June 12th, 2016
  Complete the requisite forms throughout the life of this phase.

- **Landscape View Mode (Jorge Rodriguez)** - May 1st, 2016
  Create the non-interactive 3D backdrop for the view modes.

- **Object Relationship Matrix (Jorge Rodriguez)** - May 1st, 2016
  Program the matrix against which all objects assume their relative positions and relationships with each other.

- **Landscape Edit Mode (Jorge Rodriguez)** - May 15th, 2016
  Create the non-interactive 3D backdrop for the edit/create modes.

- **3D Mesh Factory (Jorge Rodriguez)** - May 15th, 2016
  Implement the automated creation of 3D objects associated to the transpiled json string.

- **Function Graph (Jorge Rodriguez)** - May 29th, 2016
  Program the logic involved in the function graph that will attach to the GUI components in its UI counterpart.

- **Error Detection Handler (Jorge Rodriguez)** - May 29th, 2016
  Implement the catch logic for user errors, and relay relevant response to user visually/audibly.

- **Create Sound Cue Resources (Jorge Rodriguez)** - June 12th, 2016
  Develop audio assets to signal to a user that various events have taken place within VEDIC's 3D environment.

- **Implement Sound Cue Resources (Jorge Rodriguez)** - June 12th, 2016
  Place the visual cue resources in their respective event handlers.

- **Keyboard/Mouse Handler (James Vinson)** - May 1st, 2016
  Program all listeners relevant to valid keyboard keys, mouse clicks, and mouse movements.

- **UI View Mode (James Vinson)** - May 1st, 2016
  Create assets for, and implement said assets, into the view mode for database view mode.

- **UI Edit Mode (James Vinson)** - May 15th, 2016
  Create assets for, and implement said assets, into the view mode for database edit/create mode.

- **Parent/Child Adoption (James Vinson)** - May 29th, 2016
  Implement the structure to associate an object placed inside another object as child of parent.

- **Application State Manager (James Vinson)** - May 29th, 2016
  Program the transitions from one mode to the next based off of a user-instigated event.

- **Logger (James Vinson)** - May 29th, 2016
  Code and set in place the system for log creation for debugging purposes.

- **Session Storage (James Vinson)** - June 12th, 2016
  Create a saved state system for the VEDIC environment to make loadable later.

Implementation Phase - Programming (November 13th, 2016)

- **Leap Handler (William Funk)** - July 3rd, 2016
- **k**
- **Position Management System (William Funk)** - July 17th, 2016
- **k**

- **Demo C# Project (William Funk)** - July 17th, 2016
- **k**
- **Create Visual Cue Resources (William Funk)** - July 31st, 2016
- **k**
- **Implement Visual Cue Resources (William Funk)** - July 31st, 2016
- **k**
- **Parser/Scanner Input (William Funk)** - September 11th, 2016
- **k**
- **Transpiler (William Funk)** - October 30th, 2016
- **k**
- **Presentation Script (William Funk)** - November 6th, 2016
- **k**
- **Object Relationship Matrix (Jorge Rodriguez)** - July 3rd, 2016
- **k**
- **Error Detection Handler (Jorge Rodriguez)** - July 24th, 2016
- **k**
- **Function Graphs (Jorge Rodriguez)** - August 21st, 2016
- **k**
- **3D Mesh Factory (Jorge Rodriguez)** - September 11th, 2016
- **k**
- **Create Sound Cue Resources (Jorge Rodriguez)** - September 25th, 2016
- **k**
- **Implement Sound Cue Resources (Jorge Rodriguez)** - September 25th, 2016
- **k**
- **Documentation (Jorge Rodriguez)** - October 9th, 2016
- **k**
- **Presentation Montage (Jorge Rodriguez)** - October 23rd, 2016
- **k**
- **Presentation Demo Database (Jorge Rodriguez)** - November 6th, 2016
- **k**
- **Presentation C# Demo (Jorge Rodriguez)** - November 13th, 2016
- **k**
- **Keyboard/Mouse Handler (James Vinson)** - June 26th, 2016
- **k**
- **UI View Mode (James Vinson)** - June 26th, 2016
- **k**
- **UI Edit Mode (James Vinson)** - July 17th, 2016
- **k**
- **Parent/Child Adoption (James Vinson)** - August 7th, 2016
- **k**
- **Application State Manager (James Vinson)** - August 14th, 2016
- **k**
- **Logger (James Vinson)** - September 4th, 2016
- **k**
- **Session Storage (James Vinson)** - October 2nd, 2016
- **k**
- **Parser/Scanner Output (James Vinson)** - November 13th, 2016
- **k**