

Technical Design Document



Version 2.2

|  |  |  |
| --- | --- | --- |
| Position Title | Name | Signature |
| Assistant Producer | Jason Orsatti |  |
| Game Designer | Grace Blessey |  |
| Level Design Lead | Josh Weeks |  |
| Software Lead | Casey Donnellan |  |
| Art Lead | Matt Musante |  |
| Level Designer | Nick Urko |  |
| Level Designer | David Skaggs |  |
| Level Designer | Al Nachman |  |
| Level Designer | Jaddua Ross |  |
| Artist | David DeCoster |  |
| Artist | Adriana Clonts |  |
| Artist | Brian Wells |  |
| Software Engineer | Ben Pope |  |

**Document Revisions Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Description** | **Requestor** | **Date** |
| **0.1** | First Draft | Jason Orsatti | 7/1/11 |
| **1.0** | Milestone 1 Version | Professors McCoy, Rosemond | 7/5/11 |
| **2.0** | Milestone 2 Version | Professors Rosemond, Stringer | 8/22/11 |
| **2.1** | VS Version, revised based on professor feedback | Professors Rosemond, Stringer | 10/6/11 |
| **2.2** | Final Version, component updates | Professors Clune, Stringer | 12/9/11 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Table of Figures 6](#_Toc305614588)

[Project and Motivation 7](#_Toc305614589)

[Scope 7](#_Toc305614590)

[End Product 8](#_Toc305614591)

[Mechanics 9](#_Toc305614592)

[Deliverables 10](#_Toc305614593)

[System Requirements 11](#_Toc305614594)

[Target System 11](#_Toc305614595)

[Minimum 11](#_Toc305614596)

[Recommended 11](#_Toc305614597)

[Development System 11](#_Toc305614598)

[Resource Budget 12](#_Toc305614599)

[CPU Execution Time Estimate 12](#_Toc305614600)

[Memory Utilization Estimate 12](#_Toc305614601)

[Assets Budget Estimate 12](#_Toc305614602)

[Art Assets Budgets Estimate 12](#_Toc305614603)

[Acquired Technology and External Tools 13](#_Toc305614604)

[Art Creation 13](#_Toc305614605)

[Software Engineering 13](#_Toc305614606)

[Level Design 13](#_Toc305614607)

[Miscellaneous 13](#_Toc305614608)

[Component Descriptions and Risk Assessments 14](#_Toc305614609)

[Votive: 14](#_Toc305614610)

[Sacred Bulb: 14](#_Toc305614611)

[Pillbug: 14](#_Toc305614612)

[Slug: 15](#_Toc305614613)

[Snapdragon: 15](#_Toc305614614)

[Camera System: 15](#_Toc305614615)

[Checkpoint System: 15](#_Toc305614616)

[Software Configuration Management 17](#_Toc305614617)

[Naming Convention 17](#_Toc305614618)

[Category/File Formats 17](#_Toc305614619)

[Build Plan 18](#_Toc305614620)

[Backup Routines 18](#_Toc305614621)

[Version Control 18](#_Toc305614622)

[General Steps for Committing to Source Control 19](#_Toc305614623)

[SVN Client 19](#_Toc305614624)

[Checkout from the SVN 19](#_Toc305614625)

[SVN Folder Structure 19](#_Toc305614626)

[Updating from the SVN 21](#_Toc305614627)

[Adding to the SVN 21](#_Toc305614628)

[Committing to the SVN 21](#_Toc305614629)

[Resolving Conflicts 22](#_Toc305614630)

[Quality Assurance 25](#_Toc305614631)

[Component Testing 25](#_Toc305614632)

[Integration Testing 25](#_Toc305614633)

[Kleenex Test Plan 25](#_Toc305614634)

[Bug Tracking 26](#_Toc305614635)

[How to Use Issue Manager 26](#_Toc305614636)

[Arbor Questionnaire 30](#_Toc305614637)

[Tutorials 34](#_Toc305614638)

[Advanced Scripting - UnrealScript 34](#_Toc305614639)

[Anim Tree 34](#_Toc305614640)

[AI 34](#_Toc305614641)

[Basic Scripting - Kismet 34](#_Toc305614642)

[Build/Installer 34](#_Toc305614643)

[Lighting 34](#_Toc305614644)

[Materials 34](#_Toc305614645)

[Physics/Collision 34](#_Toc305614646)

[Particle Effects 34](#_Toc305614647)

[Matinee 34](#_Toc305614648)

[Terrain 34](#_Toc305614649)

[UI/HUD 34](#_Toc305614650)

[Weapon Set-up 34](#_Toc305614651)

[Sound/Music 35](#_Toc305614652)

# Table of Figures

[Figure 1- Deliverables 10](#_Toc305614653)

[Figure 2 – CPU Utilization 12](#_Toc305614654)

[Figure 3 – Memory Utilization 12](#_Toc305614655)

[Figure 4 – Art Tools 13](#_Toc305614656)

[Figure 5 – Software Engineering Tools 13](#_Toc305614657)

[Figure 6 – Level Design Tools 13](#_Toc305614658)

[Figure 7 – Components 14](#_Toc305614659)

[Figure 8 - UnrealScript Files 17](#_Toc305614660)

[Figure 9 - File Format 17](#_Toc305614661)

[Figure 10 - Naming Conventions 18](#_Toc305614662)

[Figure 11: SVN Checkout 19](#_Toc305614663)

[Figure 12: SVN Update 21](#_Toc305614664)

[Figure 13: Adding to SVN 21](#_Toc305614665)

[Figure 14: Committing to SVN 22](#_Toc305614666)

[Figure 15: SVN Conflicts 23](#_Toc305614667)

[Figure 16: Using TortoiseMerge 24](#_Toc305614668)

[Figure 17 – Issue List 26](#_Toc305614669)

[Figure 18 – Add New Issue Command 27](#_Toc305614670)

[Figure 19 – Add New Issue Dialog 27](#_Toc305614671)

[Figure 20 – Mark an Issue Command 28](#_Toc305614672)

[Figure 21 – Mark an Issue Dialog 28](#_Toc305614673)

# Project and Motivation

Arbor is a gloomy, enchanting, atmospheric 2D side-scrolling puzzle platformer with playable foreground and background zones set within the ruined Temple of the Arbor Goddess. Whatever civilization once created the massive structure has long since perished. To rejuvenate her ruined temple, the Goddess brings to life the last of a legion of tiny clay figures carved to serve her. This humble servant, simply called Votive, must brave the wastes now overrun with Dark Miasma, crumbling architecture, and corrupt creatures. He must gather and transport Sacred Bulbs to the plots of Hallowed Soil that still remain in the Temple. Only then can the Bulbs sprout and flourish, and the Goddess restore life to her quickly failing world.

The player accesses each level of the game from the central Hub, and uses platforming and puzzle-solving skills to progress forward through the levels in search of Sacred Bulbs. Upon finding a Bulb, the player carries or rolls it past more obstacles to a plot of Hallowed Soil. Planting every Bulb within the level completes the level and restores life to that section of the temple.

Votive uses the power of the three Magic Leaves that grow in the Temple to platform through the level. The Bounce Leaf shields Votive’s fragile form from damage when he jumps off the Temple’s stone walls. With the Glide Leaf, he flies across large gaps and rides up vents of air. Finally, the Vine Leaf allows Votive to swing up to high ledges and across dangerous pits. As he explores, Votive collects the Shards of other broken votive figurines so that the Goddess may return them to life someday as well.

Votive also interacts with the creatures of the Temple. Dangerous Slugs and Snapdragons kill Votive if he does not avoid them. Timid Pillbugs curl up when Votive approaches, allowing him to pick up and carry them. He uses Pillbugs to weigh down Buttons and feed Snapdragons, thus granting him access to new areas.

*Arbor* has a dark and spooky feel rendered in stylized realism. The Temple Grounds are enormous and create an intimidating atmosphere. Players feel alone and overwhelmed by both the sadness of the Temple and the beauty that must once have been.

## Scope

From a technical perspective, Arbor is not a complex game. Kismet scripts and physics drive the majority of interactions in the game. Nearly all technical development focuses on controlling Votive and refining that control. Other areas include solving corner cases where physics can break down and impede progress, enemies, and building the tools used to create levels.

Arbor uses the June 2011 version of UDK. The project largely utilizes prismatic rigid body actors to constrain the gameplay to two dimensions. One very important goal of this project is to extend the entire code base from UDK itself and not use any code from Unreal Tournament. UT3 is an excellent reference for how to write UnrealScript, but extending it, as in previous projects, actually makes things far more difficult. Deriving all components from UDK will create some extra work for the programming department, but the net benefit is still positive, as it will drastically reduce the incidence of bugs appearing from hidden dependencies within UT3.

With sixteen weeks to complete production, there is virtually no chance we will run out of time to implement features. The programming department’s goal for this project is feature completion in eight weeks. This leaves another entire eight weeks for aesthetic refinement, performance tuning, and bug squashing.

## End Product

Votive has a number of innate abilities as well as three special powers. It is capable of running, jumping, and pushing the bulb as well as picking up and tossing pillbugs. Votive can also pickup the sacred bulb itself, although its movement is severely impeded when doing so. Votive can also acquire three special abilities by picking up magic leaves spread throughout the levels. First, the glide leaf allows Votive to glide through the air, effectively slowing its fall and extending its jumps. The bounce leaf allows Votive to bounce effortlessly against floors, walls, and ceilings. Finally, the vine leaf allows Votive to launch a vine at certain locations and swing back and forth.

A number of environmental objects serve to aid Votive’s progress through the game. First, there are magical leaves growing at various points throughout the world that Votive can pick up if it is close enough. There are also doors in the background that Votive can use to travel back and forth between different layers of the game world. Air jets propel Votive in a given direction when it is using the glide ability, while grapple nodes allow Votive to use its vine to swing from one location to another.

There are other objects, however, that impede Votive’s quest as well as enemies that attempt to destroy it outright. Razor sharp spikes are plentiful throughout Votive’s world as well as deadly toxic pools. Dark Miasma, an ominous cloudlike substance robs Votive of its leaf upon contact. Evil slugs and monstrous snapdragons also stand in Votive’s way, each capable of destroying it with ease. Perhaps most dangerous of all, however, is Votive’s own fragile nature: the clay figurine can survive only the slightest of drops without shattering.

## Mechanics

Player Mechanics

* Run
* Jump
* Push
* Pickup Pillbug/Sacred Bulb
* Pickup Leaf Power
* Throw/Drop
* Glide
* Bounce
* Vine

Entities

* Snapdragon
* Sacred Bulb
* Pillbug
* Slug

Environmental Objects

* Air Jet
* Dark Miasma
* Kill Volumes
* Grapple Nodes
* Collectible Shard
* Hallowed Soil
* Doors

Other Systems

* Checkpoints
* Camera System
* HUD
* Main Menu
* Pause Menu

## Deliverables

SkyGate provides a DVD, which automatically installs *Arbor* on a user’s computer. SkyGate’s development resources, including all art, code, and level assets are available via SVN at: [**http://ghsrv.ecsrv.smu.edu/svn/c15\_Team\_skygate**](http://ghsrv.ecsrv.smu.edu/svn/c15_Team_skygate)

|  |  |
| --- | --- |
| **Asset** | **Description** |
| **Autorun.inf** | Automatically starts installer from DVD |
| **ArborSetup.exe** | Installer executable, installs to C:/UDK/Arbor by default |
| **Arbor.ico** | Installer icon |
| **Readme.txt** | Installation notes, other cogent information |
| **ArborTrailer.wmv** | Game Trailer |
| **Box** |  |
| **Poster** |  |
| **Manual** |  |

Figure 1- Deliverables

# System Requirements

## Target System

## Minimum

* ***Processor:*** Intel Pentium 4 1.3 GHz or AMD Athlon XP 1500
* ***Memory:*** 1GB or more of RAM
* ***Hard Drive:*** 25 GB free hard drive space
* ***Video Card:*** NVIDIA® GeForce® FX or ATI Radeon™ 9500 video card or better
* ***Operating System:*** Windows XP 32bit (SP3)
* ***Peripheral:***X-Box 360 Controller
* ***Sound:***  DirectX 9-compatible sound card or motherboard sound capability

## Recommended

* ***Processor:*** Dual-core processor, such as the Intel Pentium D or AMD Athlon 64 X2
* ***Memory:*** 2 GB RAM (4 GB for Vista users)
* ***Hard Drive:*** 25 GB free hard drive space
* ***Video Card:*** 3D graphics processor with Vertex and Pixel Shader capability with 256 MB NVIDIA® GeForce® 8600 or ATI Radeon™ 2600 or better
* ***Operating System:*** Windows Vista 64bit (SP1) Windows 7
* ***Peripheral:***X-Box 360 Controller
* ***Sound:*** DirectX 9-compatible sound card or motherboard sound capability

## Development System

* ***Processor:*** Intel dual-core 2.4GHz
* ***Memory:*** 4 GB RAM
* ***Hard Drive:*** 451 GB Hard Drive
* ***Video Card:*** ATI Mobility Radeon HD 5870 with 2778 MB RAM
* ***Operating System:*** Windows 7 Ultimate Edition (64-bit)
* ***Software:*** see [Technology and Tools](#_Acquired_Technology_and)***Peripheral:***X-Box 360 Controller, Mouse, Keyboard
* ***Sound:*** DirectX 9-compatible sound card or motherboard sound capability

# Resource Budget

## CPU Execution Time Estimate

|  |  |
| --- | --- |
| **System** | **Execution Time (ms)** |
| **Input** | 0.25 |
| **Gameplay** | 1 |
| **Physics** | 1 |
| **Artificial intelligence** | 0.25 |
| **Sound system** | 0.25 |
| **Animation** | 0.5 |
| **Lighting** | 1 |
| **Special effects** | 1 |
| **Rendering** | 10 |
| **Total** | 15.25 |

Figure 2 – CPU Utilization

## Memory Utilization Estimate

|  |  |
| --- | --- |
| **Subsystem** | **Memory usage** |
| **Compiled Script** | 10MB |
| **Game Level** | 200MB |
| **Models** | 50MB |
| **Textures** | 175MB |
| **Sounds** | 50MB |
| **Total** | 585MB |

Figure 3 – Memory Utilization

## Assets Budget Estimate

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Occurrence** | **Polygons (per Unit)** | **Polygons (max)** |
| **Enemies** | 0-5 | 3,000 – 8,000 | 40000 |
| **Player** | 1 | 3000-8000 | 8000 |
| **Environment Objects** | 50 – 100 | 400 – 1,000 | 100000 |
| **Particle Effects** | 0 – 50 | 200 – 400 | 20000 |
| **Total** | – | – | 168000 |

Figure 4 – Model Polygon Limits

## Art Assets Budgets Estimate

Based on stress test results for polygon counts the development team established a theoretical maximum value of 1.5 million total visible polygons per frame for 30 frames per second gameplay. This budget is simply an estimate and game performance depends on many factors other than the size and level of detail of the art assets. These are conservative estimates in order to mitigate any potential issues with maintaining a stable frame rate further into the project.

|  |  |
| --- | --- |
| **Type** | **Max** |
| **Texture Size** | 4096 x 4096 |
| **Dynamic Lights** | 5 |
| **Particles** | 4000 |

Figure 5 – Asset Limits

# Acquired Technology and External Tools

### Art Creation

|  |  |
| --- | --- |
| **Tool** | **Description** |
| **Autodesk 3DS Max 2011** | Used to create original models and animations. |
| **Autodesk Mudbox 2011** | Used to facilitate the modeling process. |
| **Adobe Photoshop CS5** | Used to create and modify textures and materials. |
| **ActorX** | Plugin for 3DSMax used to export character animations. |
| **XNormal** | Used to generate normal, ambient occlusion, and displacement maps. |

Figure 4 – Art Tools

### Software Engineering

|  |  |
| --- | --- |
| **Tool** | **Description** |
| **Microsoft *Visual Studio*** | The IDE used to develop all UnrealScript on the project. |
| **Pixel Mine nFringe** | Integrates with Visual Studio and facilitates the development of unreal script code while providing support for syntax highlighting and code navigation. |

Figure 5 – Software Engineering Tools

### Level Design

|  |  |
| --- | --- |
| **Tool** | **Description** |
| **Unreal Development Kit (June 2011)** | The game engine and framework that *Arbor* is built in. |

Figure – Level Design Tools

### Miscellaneous

|  |  |
| --- | --- |
| **Tool** | **Description** |
| **Microsoft Excel** | Used to manage asset lists and the product backlog. |
| **Tortoise SVN** | Serves as a shared, collective repository with support for sub-versioning and reversion. |
| **Fraps** | Used for video and screen capture. |
| **Hypercam 2** | Used for video and screen capture. |
| **Microsoft Word** | Document editing |
| **Issue Manager** | Bug tracking |
| **Audacity** | Sound editing |
| **Notepad** | .ini editing |

Figure 9 – Misc Tools

Components



Figure 7 – Components

## Component Descriptions and Risk Assessments

### Votive:

**Description:** *The player’s surrogate in Arbor with which he can interact with game objects and the environment. Votive has the innate ability to run, jump, and push other objects. He also can acquire the ability to bounce off walls and the ground, glide in the air, and swing around with vines.*

**Impact:** *enormous; this needs to feel just right for a fun game*

**Probability:** *medium; lots of mechanics, requires a high level of polish*

**Visibility:** *colossal; work on this begins immediately, constantly visible to entire team*

**Mitigation:** *no mitigation possible; the game does not exist without this*

**Current Status:** *12/9/11 – complete*

### Sacred Bulb:

**Description:** *A large, round physics object that can be pushed by the player and rolls around the environment. Has weight and can interact with other physics objects. This component also serves as a goal piece that the player must find and move in order to complete any level.*

**Impact:** *high; this object is a core component and part of every level*

**Probability:** *medium; physics object can be difficult to work with*

**Visibility:** *high; work on this begins immediately, constantly visible to entire team*

**Mitigation:**  *the bulb plays a lesser role in the game, moves slowly*

**Current Status:** *12/9/11 – complete*

### Pillbug:

**Description:** Patrols along a short, straight path and rolls into a small ball when Votive comes close. The rolled pillbug is a physics object that interacts with the environment and weights down buttons and other objects.

**Impact:** *medium; the pillbug is important but not a core mechanic*

**Probability:** *medium; physics object; pick up and throw mechanics are complex*

**Visibility:** *medium; work begins towards the beginning of development*

**Mitigation:** *less interaction with Votive and/or environment*

**Current Status:** *12/9/11 – complete*

### Slug:

**Description:** *Patrols along a short, straight path and destroys Votive on contact.*

**Impact:** *medium; the slug is important but not core*

**Probability:** *low; very simple mechanics*

**Visibility:** *medium; implemented towards beginning of development*

**Mitigation:** *remove slugs; implement with Kismet/matinee*

**Current Status:** *12/9/11 – complete*

### Snapdragon:

**Description:** *Immobile actor that kills votive if he passes in front. Placated by feeding a rolled up pillbug, allowing votive to pass safely.*

**Impact:** *high; important game mechanic*

**Probability:** *low; not complex, although aesthetics are vital*

**Visibility:** *medium; implemented towards beginning of development*

**Mitigation:** *remove snapdragons; implement with Kismet/matinee*

**Current Status:** *12/9/11 – complete*

### Camera System:

**Description:** *Manages the position and orientation of the game camera. Customizable by level designers to control how each section of their levels are displayed.*

**Impact:** *high; important for all aspects of game*

**Probability:** *low; not terribly complicated, highly content driven*

**Visibility:** *high; implement immediately*

**Mitigation:**  *simplify the camera system, remove configurability*

**Current Status:** *12/9/11 – complete*

### Checkpoint System:

**Description:** *Driven by placeable trigger volumes that save the game state when the player crosses them so that the level can be restored to that point upon player death.*

**Impact:** *high; critical for playability*

**Probability:** *low; relatively simple component*

**Visibility:** *high; implement immediately*

**Mitigation:**  *simplify the checkpoint system, break sections up into separate levels*

**Current Status:** *12/9/11 – complete*

**UnrealScript Files**

|  |  |
| --- | --- |
| **File Name** | **Description** |
| **A\_AirJet.uc** | Object that can be toggled to levitate Votive when overhead. |
| **A\_BounceComponent.uc** | Provides Votive with the ability to bounce off of walls and the ground. |
| **A\_BulbReplicator.uc** | Updates and resets bulb position and state so it can be properly respawned. |
| **A\_BulbSpawner.uc** | Respawns the sacred bulb during play. |
| **A\_CameraComponent.uc** | The camera used in Arbor. |
| **A\_CameraNode.uc** | Controls camera pathing during custom scene transitions. |
| **A\_CameraVolume.uc** | A volume that defines a particular scene within a level. Each camera volume has a custom viewpoint for displaying that particular section in a specific way. |
| **A\_CarryComponent.uc** | A component that allows votive to carry other objects in the game. |
| **A\_Checkpoint.uc** | Triggers a restore point for Votive and the game state. |
| **A\_Component.uc** | Base component type that other Arbor components derive from. |
| **A\_ConcatenateStrings.uc** | Used to concatenate two different strings into one object. |
| **A\_FracturedPawn.uc** | This is a fractured mesh representing Votive that can explode, used in death sequence. |
| **A\_Game.uc** | The game type used by the engine, contains high-level configuration and game management. |
| **A\_GrappleNode.uc** | An attachment point for Votive’s vine ability. |
| **A\_HUD.uc** | The HUD type used by the game and related functionality. |
| **A\_KActor.uc** | Basic physics actors that other Arbor physics actors derive from. |
| **A\_KillVolume.uc** | A trigger volume that destroys the player and other objects upon contact. |
| **A\_Miasma.uc** | A trigger volume that removes Votive’s powers as he passes through. |
| **A\_PatrollerController.uc** | A base class for AI patrollers in Arbor. Contains basic pathing functionality. |
| **A\_PatrollerPawn.uc** | Base pawn type used by all patrolling enemy classes in Arbor. |
| **A\_PatrollerSpawner.uc** | Simple spawner that all patrolling enemy spawners derive from. |
| **A\_Pawn.uc** | The basic pawn class that all pawns derive from. |
| **A\_Pillbug.uc** | Basic physics actor for a rolled-up pillbug enemy. |
| **A\_PillbugController.uc** | Handles pillbug-specific AI behavior. |
| **A\_PillbugPawn.uc** | The physical pillbug pawn that animates and interacts with the game environment. |
| **A\_PillbugReplicator.uc** | The replicator used for the pillbug class. |
| **A\_PillbugSpawner.uc** | Used to spawn or respawn pillbugs as needed during play. |
| **A\_PlayerController.uc** | The player controller class that handles player input and updates Votive. |
| **A\_RBConstraint.uc** | Arbor physics constraint. |
| **A\_RBConstraintSetup.uc** | Used to configure Arbor physics constraints. |
| **A\_Replicator.uc** | Serves as a base for other Arbor replicators. |
| **A\_SacredBulb.uc** | The physical sacred bulb object, rolls around with basic physics. |
| **A\_SeqRespawn.uc** | Kismet event signaling a level restart or checkpoint restore. |
| **A\_SlugController.uc** | Handles slug-specific AI behavior. |
| **A\_SlugPawn.uc** | The physical slug pawn that animates and interacts with the game environment. |
| **A\_SlugSpawner.uc** | The specific spawner used to create slugs during play. |
| **A\_SnapDragonController.uc** | Animates the snap dragon pawn and manages its state transitions. |
| **A\_SnapDragonPawn.uc** | The snap dragon object containing the mesh, animations, and sounds. |
| **A\_SnapDragonSpawner.uc** | Respawns the snap dragon during play as needed. |
| **A\_SwingingComponent.uc** | This component allows Arbor actors to from grapple nodes and simulates the physics involved. |
| **A\_Twig.uc** | Twigs grow leaves that provide Votive with new abilities and powers. |
| **A\_VineComponent.uc** | The vine is the version of the swinging component used by Votive. |
| **A\_VotivePawn.uc** | The game pawn representing the player in the game. |
| **A\_VotiveSoundPawn.uc** | A base pawn that the votive pawn derives from. Contains all the sounds and audio components used by Votive. |

Figure - UnrealScript Files

# Software Configuration Management

## Naming Convention

The naming convention for software development assets is simple. All files are prefixed with ‘A\_’, followed with a descriptive name in upper camel case. The name itself should indicate the UDK type the asset ultimately derives from. For example, a asset derived from UDKPawn should contain the word Pawn in the name. Configuration files should be prefixed with the entire word ‘UDKArbor’.

## Category/File Formats

|  |  |  |
| --- | --- | --- |
| **File** | **Format** | **Example** |
| **Unrealscript File** | Upper camel case, A\_ prefix | A\_VotivePawn.uc |
| **Config File** | Upper camel case,UDK Arbor prefix | UDKArbor.ini |

Figure - File Format

**Coding Standards**

SkyGate maintains a rigid but thought out coding standard. This is essential to create maintainable code that is easy to integrate among multiple programmers.

**Guidelines**

1. Do not put curly braces on the same line as if or while statements
2. Use assertions and log statements liberally
3. Extreme programming once per week
4. Comment where additional explanation is necessary
5. Nouns for variables and classes, verbs for functions
6. At all costs, do not break the build
7. Write code that does not need to be revised later

|  |  |  |
| --- | --- | --- |
| **Category** | **Convention** | **Example** |
| **Files** | A\_ followed by upper camel case | A\_MyFile.uc |
| **Local variables** | lower camel case | myVariable |
| **Normal functions, events** | lower camel case | myFunction |
| **Enumerations** | upper camel case, plural | LeafPowers |
| **Enumeration item** | enum name, underscore, upper camel | LeafPower\_Glide |
| **Struct** | upper camel case | MyStructure |
| **Exec functions** | upper camel case | SwitchLeafPower |

Figure 10 - Naming Conventions

Additional Standards: <http://udn.epicgames.com/Three/CodingStandard.html>

## Build Plan

The Build Master (Casey Donnellan) locks the build one week prior to each milestone date. The build master must approve all further changes to code, levels, and art assets. The build begins two days before each milestone date. After the build begins, the build master only implements changes affecting the viability of the build (whether or not it runs on someone else’s computer).

After completing a successful build the build master installs it and attempts to play. If this is successful, the build master attempts to repeat this process on another computer. This is repeated each time a new build is performed. If issues are encountered during this process resolving them becomes the team’s highest priority.

## Backup Routines

Each team member’s computer is equipped with SVN. Each team member saves their work frequently and commits changes to SVN after completing each task. After each build, the build master burns a backup DVD of all original code, level, and art assets. The DVD is stored in a safe place. The lead software engineer uses SVN to restore the project, should a catastrophic event occur. The backup DVD is a last resort only.

Should a team member commit data that breaks the build and it is not immediately apparent how to repair the damage, the lead software engineer will branch the SVN repository before a point the bad commit was made. Any team members incapable of resolving their technical issues will delete their local data and get a fresh checkout from SVN.

## Version Control

SkyGate maintains all development assets under source control. Doing this is useful for several reasons. First, it allows team members to simultaneously work on the same document(s) and merge them later. It also provides the invaluable ability to “roll back” the project to a previous date in order to sidestep catastrophic changes. Finally, source control provides an excellent way of seeing exactly what each person contributed to the project.

### General Steps for Committing to Source Control

1. Update before committing.
2. Resolve all conflicts.
3. Thoroughly re-test the portion of the project the commit is affecting.
4. Write a descriptive comment about the commit.

### SVN Client

SkyGate uses the 64-bit version of TortoiseSVN as the SVN client, available at <http://tortoisesvn.net>.

### Checkout from the SVN

Before using SVN for the first time, a user must first perform a full checkout of the repository. In the folder **C:\UDK**, right-click -> SVN Checkout to bring up the following dialog:

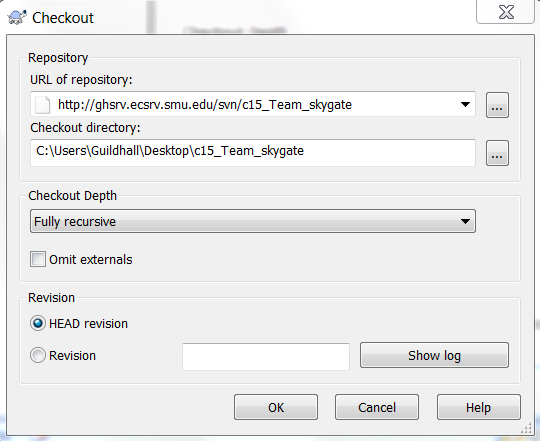


Figure 11: SVN Checkout

*SkyGate*’s SVN repository URL is:[**http://ghsrv.ecsrv.smu.edu/svn/c15\_Team\_skygate/**](http://ghsrv.ecsrv.smu.edu/svn/c15_Team_skygate/). The checkout directory should be **C:\UDK\**. The username needed to access the SVN is the first part of a team members SMU email address, and the password is the team member’s SMU student ID. If prompted by the warning that the target folder is not empty, select ‘Yes’.

### SVN Folder Structure

This details where development files are located:

|  |  |
| --- | --- |
| **Folder** | **Usage** |
| **UDK** | Base working directory |
| **UDK\Docs** | Documentation |
| **UDK\Art** | Raw art assets |
| **UDK\Audio** | Raw audio assets |
| **UDK\UDK-2011-06\Development\Src\Arbor** | *Arbor* source code |
| **UDK\UDK-2011-06\UDKGame\Config** | UDK config files |
| **UDK\UDK-2011-06\UDKGame\Content\Levels** | *Arbor* levels |
| **UDK\UDK-2011-06\UDKGame\Content\ArborPackages** | *Arbor* packages |

Table 12: SVN Folder Structure

### Updating from the SVN

To update to the latest files from the SVN, simply go to the base folder, **C:\UDK\** and right-click -> SVN Update:

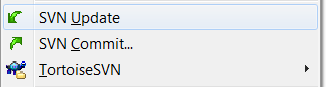


Figure 12: SVN Update

This results in a dialog box that lists all the changes since the last update.

### Adding to the SVN

Unversioned files and folders can be ‘added’ to the SVN – letting SVN know that you want to version those files sometime in the near future, but not committing (the act of actually sending them to the repository) to the SVN just yet. To do this, select the files to add, and right-click -> TortoiseSVN -> Add.

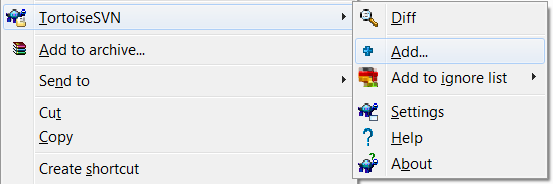


Figure 13: Adding to SVN

### Committing to the SVN

Committing is the process of actually sending your changes and added files to the SVN repository. To commit all changes across the project, go to the base folder **C:\UDK\**, and right-click -> SVN Commit. This brings up a dialog window that details all changes made, and where you can deselect items, or select previously un-added files to be committed (which has the same effect as adding and committing separately).

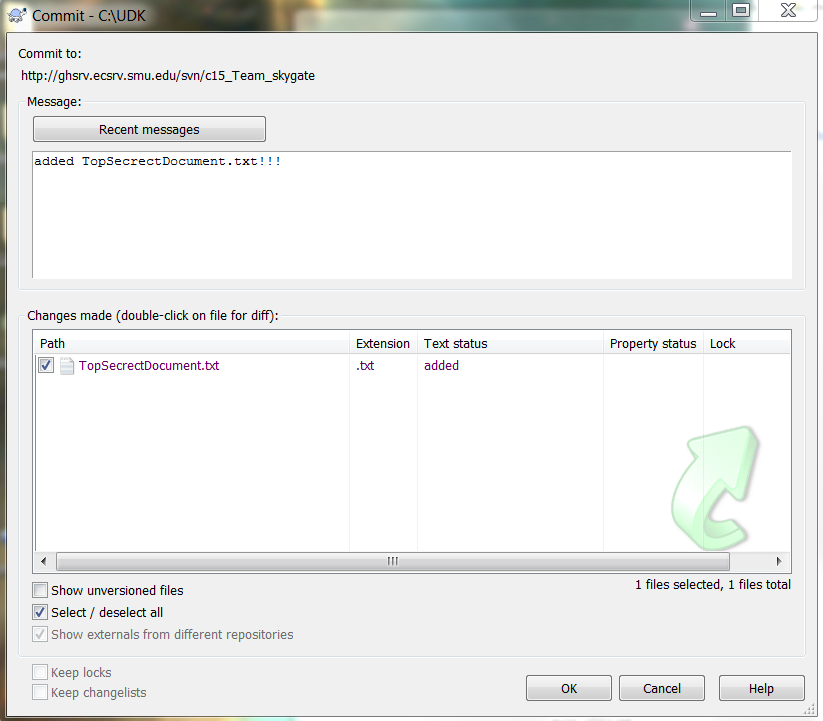


Figure 14: Committing to SVN

Enter a descriptive message detailing the commit in the dialogue box under “Recent Messages.”

### Resolving Conflicts

Conflicting files can occur from time to time when you update, and SVN is not able to figure out automatically how to merge two differing files. SVN creates three files when this happens:

* <filename>.<ext>.mine: the file with your modifications
* <filename>.<ext>.r<older revision #>: the file before you made modifications
* <filename>.<ext>.r<newer revision #>: the file that is currently on the SVN, and has a conflict with your file

To figure out how to resolve the conflict, right click on the conflicted file -> Edit conflicts.

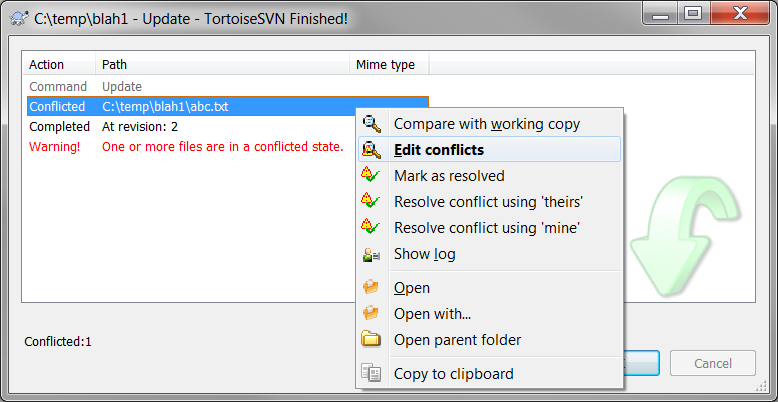


Figure 15: SVN Conflicts

This brings up a tool called TortoiseMerge, which can help visually compare the conflicting files. Use TortoiseMerge to merge the two files manually, or choose to discard either the local version or the version on the SVN. However, before attempting this, consult with the person to whom the conflicted file belongs.

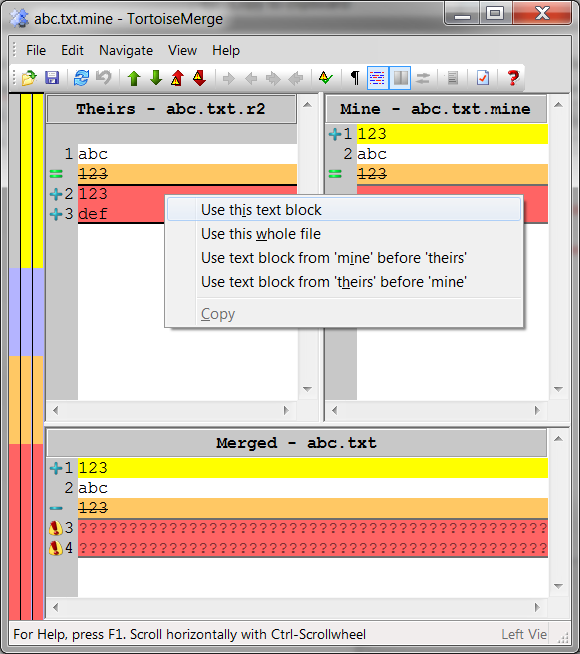


Figure 16: Using TortoiseMerge

There are several choices to make when using TortoiseMerge to resolve conflicts. By selecting a text block in either the left or the right pane and right clicking, you can opt to use that text block as the correct version, or insert it before or after the conflicting text block. You can also edit the final version manually by typing into the lower pane, titled ‘Merged - <file>.<ext>’.

After editing the conflict; save the merged version with Ctrl+S. Before marking the conflict as resolved, be certain the changes made are correct, as the new file overwrites and takes the place of the old file! Finally, select from the menu bar: Edit -> Mark as resolved. You can then close TortoiseMerge and proceed to committing the new, edited version.

If ever in doubt about merging files and resolving conflicts, consult with a member of the programming team or the Lead Software Engineer for advice.

## Quality Assurance

### Component Testing

Each individual must test their work to the best of their ability prior to integration with the main build. Each component must be functional to specification and meet project quality standards.

### Integration Testing

After successfully testing individual components individual, begin integrating work with the main project. Do not integrate during a lock down. The [Build Plan](#_Build_Plan) section of Software Configuration Management contains a description of a lock down. First, the individual must update the local working repository to synchronize with the latest build. If updates were present, the user should repeat integration testing. Once fully synchronized and component testing completes, the work uploads to the main build. Developers should label each commit comment with their name and describe the contents of the commit. Example: “Donnellan, modified bulb density.”

To ensure everything is working and integrated smoothly a daily smoke test runs at the beginning of core hours. If problems arise, the build may be temporarily reverted for the day so that individuals may continue to work.

### Kleenex Test Plan

Development uses Kleenex testing after the Vertical Slice and Alpha milestones. During testing, the team observes how the user experiences the game without instruction or interference. The team members administering the test take notes and give the tester a questionnaire following the play testing session. Developers accumulate the notes and questionnaires and hold occasional design meetings to discuss changes.

## Bug Tracking

Issue Manager is a browser-based bug tracking database used by team SkyGate to store, manage, and track the progress of all known project issues. All team members must individually enter new issues into the database as they encounter them and update them as well as when new information becomes available.

Both the producer and the lead engineer manage the database. When a new issue enters the database, the user immediately assigns it to the appropriate individual for resolving that issue. If the assignment is not obvious, a user may temporarily assign an issue to either the producer or the lead engineer. Issue Manager automatically e-mails team members in the database when a bug is assigned to them. No issue may be closed out within the database until the solution is tested and confirmed. Issues and bugs are resolved in order of priority, from highest to lowest.

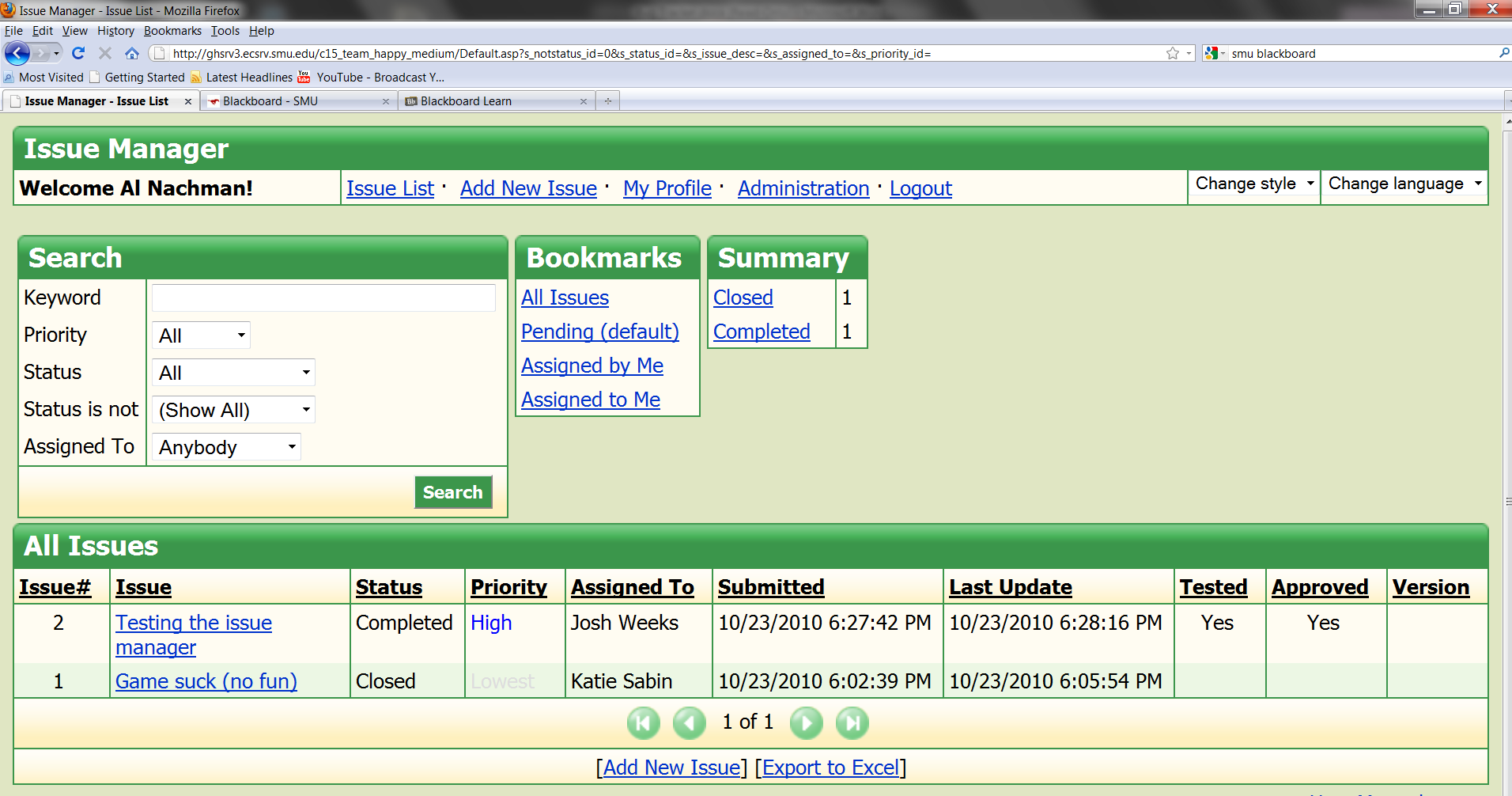


Figure – Issue List

### How to Use Issue Manager

#### Access the Issue Manager

Navigate your web browser to: [*http://ghsrv3.ecsrv.smu.edu/c15\_Team\_skygate/*](http://ghsrv3.ecsrv.smu.edu/c15_Team_skygate/)

The username is first part of your SMU email. The password is your SMU ID#.

#### View Your Issues

Select the “Assigned to me” link from the Bookmarks box. A table will appear showing all issues that are assigned to you.

#### Add a New Issue

1. Select the “Add New Issue” link.

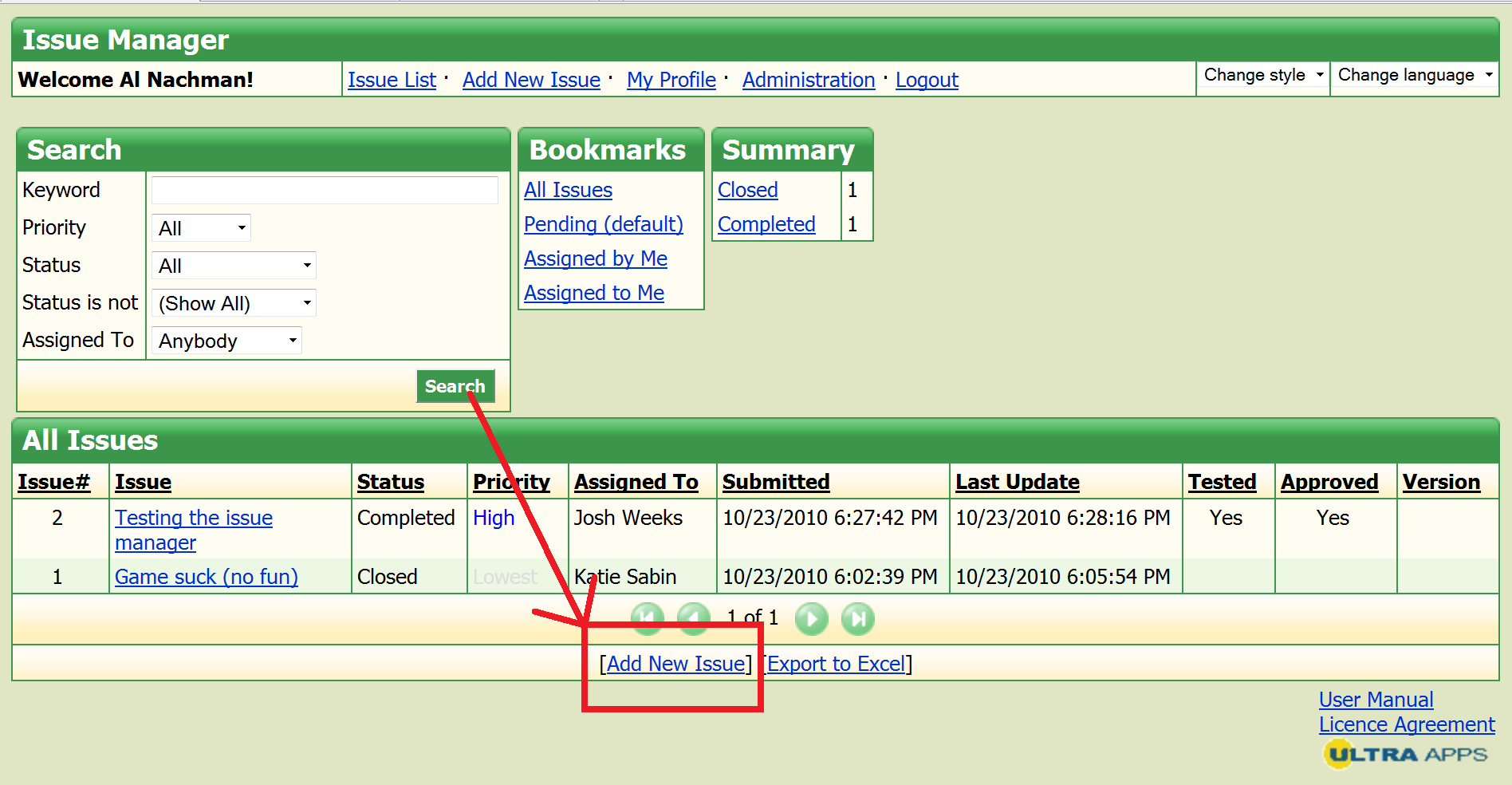


Figure – Add New Issue Command

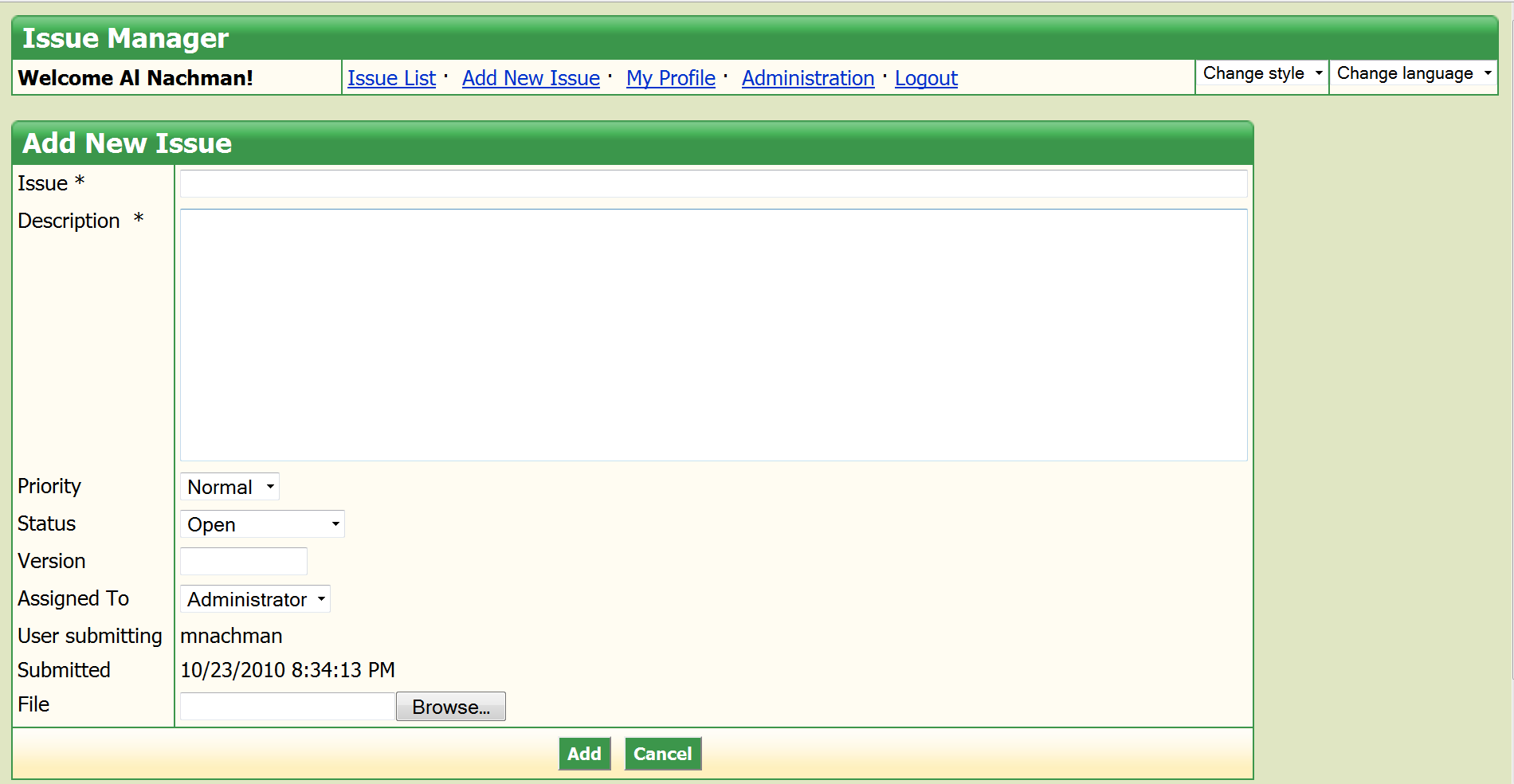
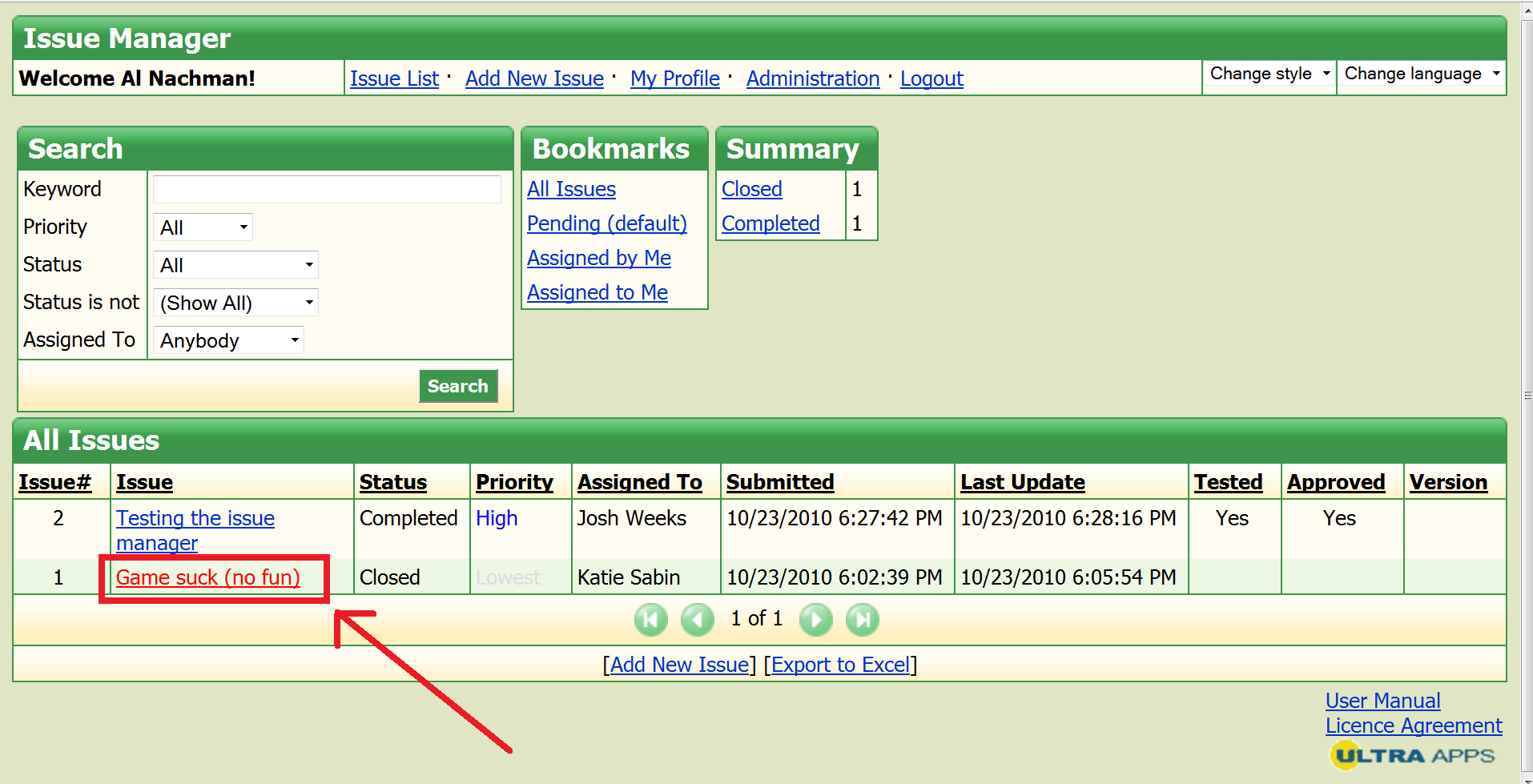


Figure – Add New Issue Dialog

1. Fill out the Issue and Description boxes when reporting the issue.
2. Assign a priority.
3. Leave Status as “Open”.
4. Use the latest build as your version number.
5. Assign the bug to the appropriate person on the list.
6. Attach a relevant file if needed.
7. Click “Add” to add the bug to the database.

#### Mark Your Issue

To mark an existing issue, click its link in the list on the main page.

Figure – Mark an Issue Command

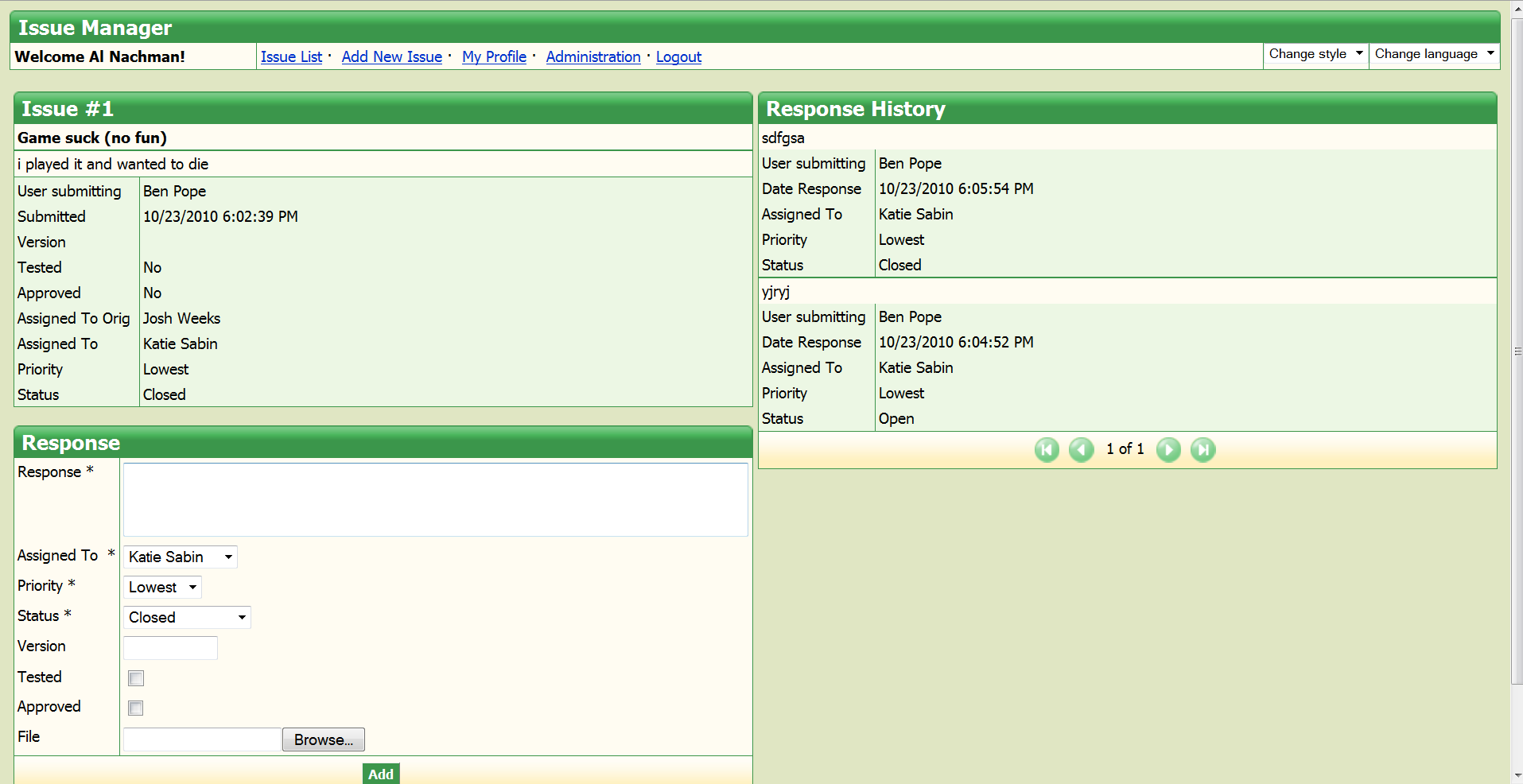


Figure – Mark an Issue Dialog

From here, other uses can modify the issue’s priority or assigned, test the issue, approve the fix, provide a response, or add a new file.

#### Close an Issue

Close an open issue by changing its status to “Closed.” Team members must get approval from the Producer or Lead Engineer to verify that the issue is resolved before closing the issue.

### Arbor Questionnaire

1. Was the game fun? Y/N:
2. What parts of the game were most fun for you?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. On a scale of 1 to 10, (with 1 being the lowest and 10 being the highest) how difficult was this game?
2. Did you feel the controls were easy to use and understand? Y/N:
   1. If you did not find the controls to be easy to understand, please explain why.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Could you relate to the main character, Votive? Y/N:
2. Did you understand the basic story conveyed through playing the game? Y/N:
   1. Please tell us what you thought the story is about.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Did you feel that the game and story had a higher meaning? Y/N:
   1. What do you think the higher meaning is about?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. If you could change anything about the game, what would it be?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Did you ever find the bonus level? Y/N:
2. Were there any moments that felt scary or intense to you? Y/N:
   1. If so, which ones made you feel that way?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What did you think about the environment?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What was your favorite enemy?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What was the most memorable part of the level?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Did you ever become stuck and not know how to continue? Y/N:
   1. If so, where?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What was your favorite leaf power?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What was your least favorite leaf power?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. On the scale of cute to horrifying ( 1 to 10 ) what was your opinion of the Snapdragon?
2. Did you find any of the in-game animations jarring or not natural? Y/N:
   1. If so, please explain where and why.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What are your final thoughts on the game; if there are any suggestions for improvements please list them here.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 

# Tutorials

## Advanced Scripting - UnrealScript

[*\\ghsrv\users\C15\TGP2\Team02-Unrealscript*](file:///\\ghsrv\users\C15\TGP2\Team02-Unrealscript)

## Anim Tree

[*\\ghsrv\users\C15\TGP2\Team04-AnimTree*](file:///\\ghsrv\users\C15\TGP2\Team04-AnimTree)

## AI

[*\\ghsrv\users\C15\TGP2\Team05-AI*](file:///\\ghsrv\users\C15\TGP2\Team05-AI)

## Basic Scripting - Kismet

[*\\ghsrv\users\C15\TGP2\Team03-Kismet*](file:///\\ghsrv\users\C15\TGP2\Team03-Kismet)

## Build/Installer

[*\\ghsrv\users\C15\TGP2\Team01-UPK-Build-Installer-SVN*](file:///\\ghsrv\users\C15\TGP2\Team01-UPK-Build-Installer-SVN)

## Lighting

[*\\ghsrv\users\C15\TGP2\Team11-Lighting*](file:///\\ghsrv\users\C15\TGP2\Team11-Lighting)

## Materials

[*\\ghsrv\users\C15\TGP2\Team12-Material\_Editor*](file:///\\ghsrv\users\C15\TGP2\Team12-Material_Editor)

## Physics/Collision

[*\\ghsrv\users\C15\TGP2\Team14-Physics\_and\_Collision*](file:///\\ghsrv\users\C15\TGP2\Team14-Physics_and_Collision)

## Particle Effects

[*\\ghsrv\users\C15\TGP2\Team09-Particle\_Editor*](file:///\\ghsrv\users\C15\TGP2\Team09-Particle_Editor)

## Matinee

[*\\ghsrv\users\C15\TGP2\Team10-Matinee*](file:///\\ghsrv\users\C15\TGP2\Team10-Matinee)

## Terrain

[*\\ghsrv\users\C15\TGP2\Team13-Terrain Editor*](file:///\\ghsrv\users\C15\TGP2\Team13-Terrain%20Editor)

## UI/HUD

[*\\ghsrv\users\C15\TGP2\Team08-Menus\_HUD*](file:///\\ghsrv\users\C15\TGP2\Team08-Menus_HUD)

## Weapon Set-up

[*\\ghsrv\users\C15\TGP2\Team06-Weapon\_Setup*](file:///\\ghsrv\users\C15\TGP2\Team06-Weapon_Setup)

## Sound/Music

[*\\ghsrv\users\C15\TGP2\Team15-Sound and Music*](file:///\\ghsrv\users\C15\TGP2\Team15-Sound%20and%20Music)