Nomon

DS4 Project Plan

Sign off:

Name	**Date**
Adam Bielinski	2011/09/26
Shaughn Perrozzino	2011/09/26
Sarah Marinoff	2011/09/26
Thomas Kenny	2011/09/26
Nik Schultz	2011/09/26

Contents

1. Project Idea

Our game, Nomon, uses light-based mechanics to create interesting puzzles in which you must alter your environment to escape grave perils. In the same way that the Greek word “Gnomon” (the arm on the sundial) means “that which reveals”, much of our game involves revealing safe paths using our light mechanics. Our project name is also a palindrome which further reflects the duality of the gameplay.

The main character, is a being of shadow. He is invulnerable as long as he is cloaked in his element. As he progresses through the game, he has to complete the puzzles in one direction. Upon reaching the end of his journey, his element is then reversed and the player has to retrace their steps through the game using the inverse shadow dynamic. The player is forced to see the world from both perspectives.

Genre & Gameplay

The game will be in the puzzle platforming genre. The player will encounter obstacles such as spiked floors, impossible jumps, etc., and overcome them by manipulating the lights and shadows in the vicinity. Objects will react to the presence of light or lack thereof, changing positions, becoming corporeal or incorporeal, etc., so the player may navigate successfully through the environment. Possible gameplay elements include:

- Light modifying devices (be they candles, or orbs that emit light/darkness)
- Spikes and other obstacles (harmful in your vulnerability, harmless in your element)
- Surfaces that become corporeal in one condition and vanish when lit/unlit
- Enemies that slumber and awaken angrily when disturbed by light
- Enemies that are frozen in light
- Objects can be moved to create shadows to hide in
- Pressure activated switches (by the player, or moved object)
- Mirrors can be moved to reflect light into shadowed regions
- Light activated switches

Target Audience & Platform

Although some BIT Students find light offensive, other people will find no objectionable content in the game. The game will have a simple control style, and as such will be at least accessible to the average user. While the puzzle mechanics might be a little difficult for very young children, there is no minimum age within reason.

We are using the Unreal Development kit; as such the main platform will be Microsoft Windows PCs and possibly the Microsoft Xbox360.

Competition

- XBLA (Xbox Live Arcade) Games
    - There is a reasonably large selection of games on the XBLA, it will be challenging to set our game apart from the others due to sheer number.
    - There are several games that fall into the Puzzle/Platformer genre as well, though they do not use the same puzzle dynamics as our concept:

- Lab Rabbit
    - A simple-style puzzle game
- The Impossible Game
    - A challenging platforming game
- Limbo
    - Puzzle adventure platformer. Heavy use of physics. Silhouettes are a major part of the aesthetics.
- Braid
    - A now somewhat famous puzzle platformer with innovative problem solving mechanics, e.g. time flow manipulation

- Flash games
    - The platformer and puzzles genres are very popular among flash game developers.
    - We have the benefit of having a 3D engine, which flash games can rarely boast.

1.1 Requirements

Structure
The game will be built using the Unreal Development Kit (August 2011 Beta). We'll be extending the Platformer Development Kit to fit our needs. More research into the particulars of coding within the UDK is needed.

Art
Following the concept of light and shadow, the colour palette of the game will be mostly monochromatic, with low saturated colour touches to maintain visual interest. Models will be simple and polished, with an emphasis on basic shapes and symbolism. They will be created in Maya 2011 or Blender3D. To create a smooth look, shaders will likely be used in place of textures wherever appropriate. The player character, environments, puzzle components and enemies will all be fully 3D creations, with animations as necessary. Special attention will be paid to lighting and particle effects, to give the game a sense of dazzle and mystic energy.

Content Organization
We're using Git for the management of our resources.

Audio
We hope to immerse the player deep into the world of light and shadow. The emphasis for the soundtrack will be on moody ambient quality, bass-heavy songs with percussive accents. The background music will either be acquired via copyright-free resources or requisitioned from a third party. Sound effects, such as user feedback for player actions, will be used minimally to avoid annoying the player.

1.2 Deliverables

The following items will be produced:

1. Project Plan
2. Design Document
3. Promotional website (currently at http://ds4.abielinski.com)
4. Promotional print materials & other advertisements (3)
5. Game assets & executables
6. Complete public source code (currently at https://github.com/windsurfer/Nomon)

2. Team Structure

	Story	Level Design	Game Architecture	AI	2D Assets	3D Assets	Audio Assets	Web Development
Adam Bielinski		X	X	X		X	X	
Nik Schultz		X	X				X	
Sarah Marinoff	X	X		X	X	X		X
Shaughn Perrozzino	X	X			X	X		X
Tom Kenny		X			X	X	X	

3. Tasks

A. Game Logic

- A-1. UDK scripting
- A-2. Level scripting
- A-3. Puzzle scripting

B. Game Asset Development

- B-1. 3D models
  - B-1.1. Character
  - B-1.2. Environment
  - B-1.3. Puzzle pieces
  - B-1.4. Shaders
- B-2. Animations
  - B-2.1. Character
  - B-2.2. Environment

- B-2.3. Puzzle pieces
- B-3. Menu items
  - B-3.1. Start screen
  - B-3.2. Pause/quit screen
  - B-3.2. Level selection screen
- B-4. Audio
  - B-4.1. Background music
  - B-4.2. Sound effects

C. Web Development

- C-1. Iterative web design
  - C-1.1. 2D assets
    - C-1.1.1. Banner/logo
    - C-1.1.2. Screenshots
    - C-1.1.3. Game art
  - C-1.2. Video assets
    - C-1.2.1. Video recording
    - C-1.2.2. Video editing
    - C-1.2.3. Audio editing
- C-2. Iterative scripting
- C-3. Iterative cross-browser compatibility testing

D. Promotional Material Development

- D-1. Posters
  - D-1.1. Design
  - D-1.2. Implementation
- D-1. Social Media

E. Testing

- E-1. Initial Testing
  - E-1.1. Test All Levels
- E-2. Alpha Testing
  - E-2.1. Recruit alpha testers
  - E-2.2. Develop testing plan
  - E-2.3. Run alpha test
  - E-2.4. Organize results
  - E-2.5. Prioritize bugs
- E-3. Beta Testing

4. Schedule & Resources

4.1 Milestones

Design Document and Advertisement - October 11th 2011
A detailed design document will be created, specifying all the information required to build the game from a design perspective. The initial game advertisements will be finalized, ready to distribute.

Gameplay Prototype - October 14th 2011
A demo game with basic elements using the Unreal Engine has been completed. This will be used as the basis for what types of puzzles can be created for what amount of effort. Not all gameplay mechanics will be implemented.

Puzzle Concepts Finalized - October 17th 2011
A detailed list of all puzzle components or “puzzle pieces” that can be combined to build a level. Assets and special programming required will be identified at this time. The solutions to the puzzles will be included.

Level Design Complete - October 26th 2011
All levels have been conceptualized and planned out. Grid-based mock-ups have been created and approved. Level designs indicate which “puzzle pieces” will be incorporated, so special requirements can be identified. The levels will be built in tandem, as designs are finalized.

Ready for Alpha Testing - November 4th 2011
All levels have been built. All placeholder assets have been incorporated into the levels, as well as many finalized assets. The player should be able to complete a full play through of the game from start to finish. An alpha testing plan has been devised and a group of testers has signed up.

Alpha Test, Advertisement and Improved Website - November 15th 2011
By mid-November, a full round of alpha testing will be finished and the results recorded and organized. The list of issues gleaned from alpha testing will be prioritized and ready to tackle. The website will be updated to include recent game content (images and videos). The second installment of advertisements will be complete and ready to distribute.

Asset Development Complete - November 30th 2011
All 3D models, 2D textures (as needed), soundtracks and sound effects have been finalized and incorporated into the game.

Final Release - December 16th 2011
The polished game is completed and fully playable, with no bugs. All support media is printed; all viral media is online and public. The website design and contents have been finalized. It is ready to distribute.

4.2 Resources

Software Development Tools
- Unreal Development Kit (August 2011 Beta)

Asset Development Tools
- Autodesk Maya 2011
- Blender3D
- Adobe Photoshop
- Adobe Illustrator
- Audacity
- Fruity Loops

Project Management Tools
- Redmine
- Git/Gitosis

5. Risks and Contingency Plan

5.1 Risk Analysis

Time Management
There are many projects underway this semester. As such, it will be difficult to allocate sufficient time for everything. The project scope may have to be reduced if issues in other projects arise that take up more time.

To avoid this, milestones have been scheduled in advance so the team members can effectively plan out what will be worked on when. If time management does become an issue, responsibilities may have to be shifted towards members with a lighter workload. If no team member is available to finish something, a feature will have to be cut.

Technical Issues
No one in the group has any prior experience working with the Unreal Engine. Unexpected problems are likely to come up and solutions may take a long time to implement. With such a short development window, even a short delay will be a major setback.

To avoid this, the team will pay avid attention to the Unreal Engine tutorials provided by Prof. Ali Arya. If the team does encounter technical issues, help will be sought from more knowledgeable individuals. In the worst case, the game may be simplified to accommodate the team's capabilities.

Animation
No one in the group has any prior experience with rigging and skinning. Though this is being taught simultaneously in another course, it is likely to be a challenging process. Due to the lack of experience, any issues that come up may take a very long time to fix, or even result in a complete loss of work.

To avoid this, the team will create simple models with simple skeletons that should be as straightforward as possible to animate. If animation becomes a big issue, the range of animations will be reduced and the remaining actions will undergo a more thorough polishing process.

Team Morale
Given the heavy workload, it will be difficult to keep the team morale high and stay motivated. Lack of motivation or interest in the project can result in procrastination or lack of communication, at which point deadlines might be missed.

To avoid this, the team will establish a good work/life balance, making sure to take breaks when necessary and try not to get burned out. The team will focus on one milestone at a time, in order to feel satisfied when a goal is accomplished. If team morale drops, a night off will be planned for the team to have a few drinks and get excited again. If communication starts to taper off, members will be confronted at school or during the work sessions of other projects.

Hardware Issues
The computers in both lab 234 and 236 run the example platformer game (in the editor) at less than 5 frames per second, making them unplayable. Our game will likely have similar geometry and shaders, so using the lab computers will not be possible to develop the game. A compiled game will achieve higher frame rates, but it is not guaranteed to be sufficiently playable.

If this is the case, we will have to tweak our game's geometry and/or lights and/or shaders to be less resource intensive. This will reduce the visual quality but render the game playable at least on the school equipment.

If our attempts at reducing resource usage fail, we will be forced to simply run the game on our personal systems, record video, and take screenshots. The code and binaries will still be publicly available for others to play.

5.2 Risk Exposure

Identified Risks	Probability (P)	Impact (I) (out of 10)	Exposure E = (I * P)
Time Management	0.8	7	5.6
Technical Issues	0.4	5	2.0
Animation	0.4	3	1.2
Team Morale	0.6	6	3.6
Hardware Issues	0.5	4	2.0

6. Quality Control and Communication Policies

6.1 Quality Control

Every project needs a way of reviewing the products and making sure they meet quality requirements. For a game, this can be done through different developer and user tests. For a web design or animation project, quality control can be reviewing page layouts, assets, animations, etc and ensuring they meet the required quality and functionality, follow the selected style, and have no forgotten parts.

We have a "definition of done" checklist for a feature. This will increase cooperation between team members, build trust, and keep everyone accountable. The checklist would be flexible and will undoubtedly change as the project progresses. The items should be obvious and not subjective, i.e. "code functions as designed" is better than "API is usable" since usable is a subjective term. An example and proposed checklist is as follows:

- Code functions as desired
- All assets are provided in source and required formats
- All required documentation is provided
- The feature is approved by the person who has the feature as their primary role

The Definition of Done document will be posted on our internal Wiki and will be subject to revisions as the project progresses.

Dealing with Issue Resolution

Upon completing and issue,

- Set issue status to resolved
- Notify the lead (for the section)
- Lead closes the issue

This checklist is not complete and merely proposed at this time. The checklist will be changed and re-evaluated as the project progresses.

Failure to meet the "definition of done" checklist for a particular feature means it is incomplete. This means that even if a feature works, even if it has been approved, and even if it has all the required art assets provided correctly, for example, it may still be incomplete if the specified documentation hasn't been provided.

A task-specific "definition of done" checklist will be provided for every single task in Redmine's issue tracker, to be selected from a longer list of possible requirements. Since one of the project's main goals is superb quality, QA will be taken very seriously for the duration of the project.

6.2 Communication Policies

In order to maintain effective communication in such a large group, a variety of communication methods have been established:

Redmine
The team's primary method of communication and documentation is a browser-based application called Redmine. The Redmine interface consists of the following key elements:

- **News**
  A digital bulletin board where members can post news items and comment on existing news items. This is where meetings are announced and scheduled as well as other important issues that need to be made aware to the team.
- **Wiki**
  A standard wiki plug-in, used for creating and storing group documents, brainstorming ideas and features, and keeping all project-related information accessible and organized. All wiki articles can be locked so that no issues will arise when two or more people are editing the document.
- **Issue Tracking**
  A hierarchical task breakdown, which tracks details such as assigned team member, work estimate, hours of work done, stage and percentage of completeness, priority, start and due dates, etc. This comprehensive component is the go-to section for all project-related tasks, and contains built-in features such as an auto-generated Gantt chart.

Issue tracking uses the following task statuses, which will help keep the team organized and aware of project progress:

Status	Description
Not Started	The issue has been created, but it is not yet time to work on it.
In Progress	Work has begun on the issue, but it is not finished yet.
Feedback	Work has begun on the issue, but the person working on the issue is blocked and needs assistance.
Resolved	Work on the issue is completed, and awaiting the section lead to sign off for QA purposes.
Revising	The section lead has reviewed the work completed and found a bug, or deemed it of insufficient quality to be finished.
Closed	The section lead has reviewed the work completed and found it to be of acceptable quality.
Late	The issue has not been completed on time.

- **Gantt Chart**
  As mentioned above Redmine is able to take our issues lists and turn the into a workable Gantt chart. This allows us to easily manage and track our progress throughout the development cycle.

Other resources will be used in addition to Redmine:

Email
Redmine is closely tied in to email, in that most Redmine activity - updating tasks, posting news, editing the wiki, etc. - will sent out an email notification. Thus, email serves as a secondary communication method, to make group members aware that changes and events are happening.

Meetings
The team has agreed to work during the scheduled lab time, as well as Monday afternoons as necessary. Lengthier work sessions will focus on pair programming, asset development and other actual deliverable work, while shorter impromptu meetings may occur throughout the week.

Cell phones
Phone calls and texting will be used for quick, informal communications, such as confirming meeting locations or notifying the team if an individual will be late.

Online Instant Messaging
Chat clients like Google Chat, Windows Live Messenger, Adium, Pidgin, etc., are useful for two reasons. During work sessions, links to online resources can be shared quickly by copy/pasting

into chat clients. If a team member has to work remotely, chat clients can be used to stay in touch for the duration of the work session.

VOIP

Mumble, Skype and other VOIP clients can be used if team members are working remotely and do not want to interrupt work-flow by switching between their task and a chat client. This is especially useful if team members are pair programming remotely.