MLC-WE05

A First-Person Perspective Virtual Tour

A Major Qualifying Project Report:

submitted to the Faculty

of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Michael P. Lundy

Tim Walsh

Date: April 27, 2006

Approved:

1. Virtual Tour
2. Serious Games
3. WPI

Professor Mark Claypool

Abstract

This idea behind this project was to create an entertaining game that could help prospective students to learn about the campus of WPI. We developed a first person perspective virtual tour based on pictures of the WPI campus. To evaluate the usefulness of the game, a user study was conducted. The user study showed us that the tour aspect of the game informed people of spatial relations of buildings on campus and the paths that connect them.

Contents

List of Figures

List of Tables

1 INTRODUCTION

People want to remotely learn about a physical space to avoid the cost of travel or because they cannot travel to the physical location. Virtual tours bring a physical space to remote users and provide an interactive view. Hotels and universities such as Worcester Polytechnic Institute (WPI) uses this tool to give patrons or prospective students insight. Virtual tours can be made using virtual environments, videos with limited interactivity or VRML views. Virtual environments bring the user into the tour and give them a sense of presence.This sense of presence provides the opportunity to teach while entertaining. This aspect would be essential to any teaching game.

The goal of this project is to create an interactive first-person virtual tour of the campus of Worcester Polytechnic Institute (WPI). The development of this tour incorporates still images of the campus and a method of navigating the virtual environment (VE). Utilizing images of the streets and walkways on the campus provides a natural way to see the campus and the pathways students commonly use. Movement is accomplished by a simple point and click interface with directional arrows that allows inexperienced users to easily navigate the tour.

Since the focus is teaching students about WPI, indications of how the campus looks and the facilities that are available to students are important aspects of the tour. Another aspect of the project is an easy method of delivery. One restriction is that the whole application would need to fit on a CD or DVD to be mailed or hand to those who are interested. Also providing a copy as part of New Student Orientation packages allow students know how to locate their classrooms and labs, and find the quickest way to get to the building they are in.

A user study to evaluate how well users learned about the spatial relations, path selection and general knowledge was used. Demographical information was collected for the purpose of eliminating people with a large degree of knowledge about WPI. The study was distributed through e-mail to four high schools in Massachusetts and New

York.

After the user study was complete we found that users showed improvement in the relationship between the location of buildings. They also showed improvement in choosing the correct path between the buildings. The study also found that the tour did not help user to learn general knowledge about WPI such as the majors offered.

This tour has the added benefit of providing a better view of the institute then campus brochures because the player can both interact with objects and focus on areas of their interest, providing a more personal outlook.

Next, in Chapter 2, we will discuss work done in virtual environments to increase the feeling of being part of the environment and the evaluation of environments. Chapter 3 discusses how to create a virtual tour, problems we faced and the solutions we used to solve them. After this we discuss, in Chapter 4, the results from the user survey and then, in Chapters 5 and 6, our finding and future work to improve the results found.

2 Related Work

Virtual environments (VE) are computer generated realities used in entertainment and educational software. There are many variations of VE such as virtual tours, game environments and virtual reality. Virtual tours are used by colleges and other businesses, such as hotels, to provide a view of either the campus or rooms from remote computers. This type of VE helps the user to understand the layout and look of the environment prior to committing themselves to visiting the physical location. Computer game and other designers create imagined worlds such as *Ages of Myst* in Cyan's game *Myst*. In these VEs, an idea or concept is used to make part or all of VE. The primary purpose of these VE's is to provide entertainment to the user.

Virtual tours, such as those offered by colleges and universities provide prospective students with a wealth of information including facilities that are open to students and an overview of what the campus looks like. An increasing number of high school seniors are using these tours to gain an insight into perspective schools [1]. As the goal of any tour application is to increase the presence, information gathering, and spatial awareness the designers must create an immersive environment that realizes this goal. This task can be broken into two parts: *presence* and *usability*.

2.0.1 Presence

Presence in a virtual environment is the feeling of being part of the environment. One of the most common interactions in a VE is travel which, if handled correctly, can help increase presence. There is no one method of travel that has been determined to always provide the maximum level of presence in a given VE. A single method of travel has not been mapped as the best method for a given type of VE. As choosing the method of travel has not been made into a deterministic science, [3] produced a taxonomy of travel techniques, which is displayed in Figure 2.1 on the following page. This taxonomy breaks a technique into three components, described in Table 2.1.

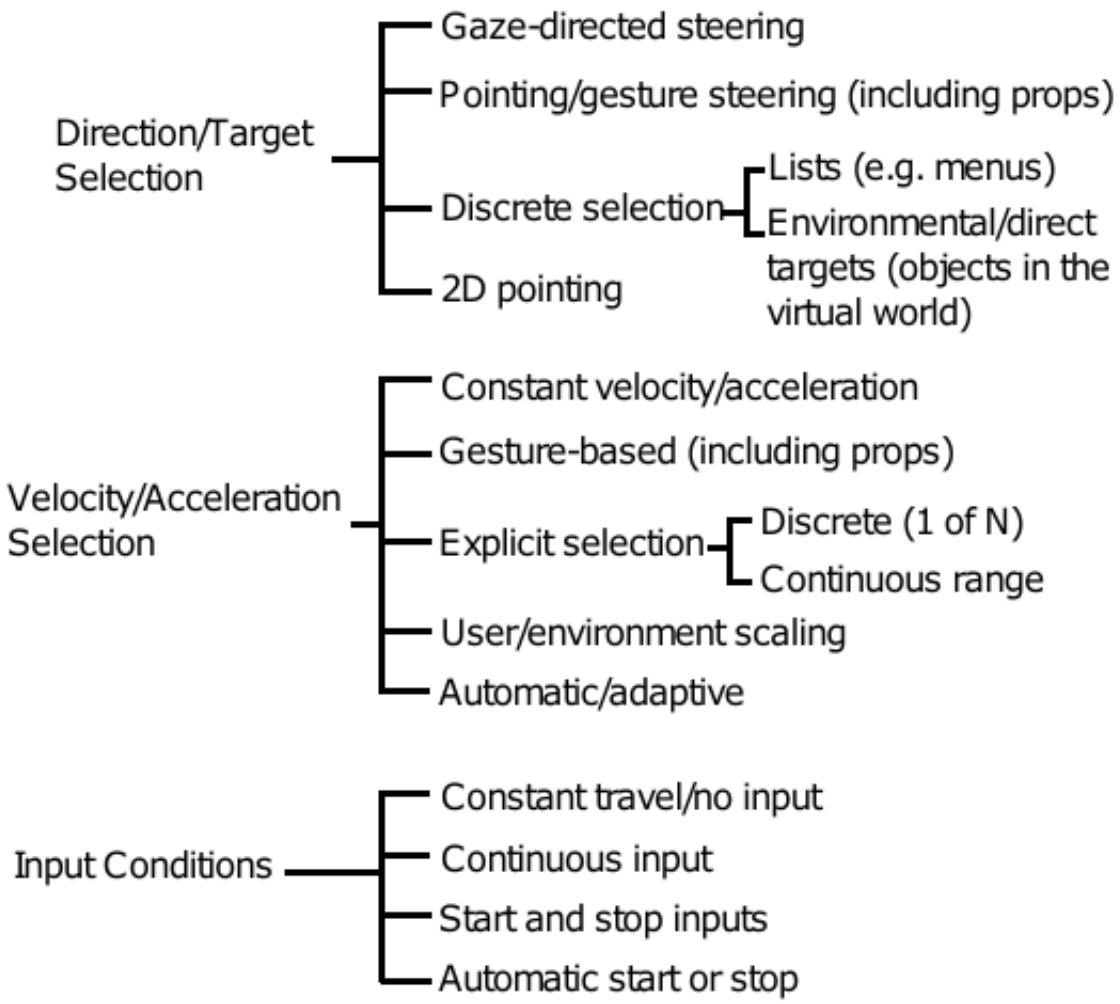


Figure 2.1: Taxonomy of travel techniques for immersive virtual environments [3]

Component	Description
Direction/Target Selection	The method used to specify the direction or object of travel
Velocity/Acceleration Selection	Provides the user with a mechanism to vary the rate travel occurs
Conditions of Input	Input required to begin, continue, and end travel

Table 2.1: Components of a travel technique

Along with the taxonomy, they made a set of eight quality factors to measure the performance of a technique. These factors are speed, accuracy, spatial awareness, ease of learning, ease of use, information gathering, presence and user comfort. Relating factors to travel techniques identifies which techniques meet the needs of the application.

A designer can apply the taxonomy to these quality factors designing a technique suitable for the requirements. Experiments done by other designers can not only be useful in designing a technique but can sometimes provide a pre-designed technique which can be adapted to a new application. As an example, [3] found that a technique using "environmental target selection with gesture-based velocity selection, explicit start inputs, and explicit or automatic stop input" would be a natural fit for an immersive tour application.

The point-and-click interface used in the tour can be analyzed by the taxonomy. This style of travel has start input, automatically stops and no continuous travel. It also has constant velocity, no acceleration and uses 2 dimensional point to navigate. Relating speed, accuracy and ease of learning and use to the point-and-click technique would determine if this technique is appropriate.

2.0.2 Usability

The next part of the design challenge is usability which is defined as the combination of ease of use and usefulness. Evaluating usability requires metrics such as learnability and user task performance. However the evaluation of usability of VEs is unlike the evaluation of usability of traditional graphical user interfaces (GUI) [4]. One reason for the difference is that some VEs must account for physical limitations, such as the user running into a wall or leaving the sensing area of the equipment used to run the VE. These VEs are generally map movement within the environment to actual movements of the user. Traditional GUIs do not require this as an evaluation criterion. VEs also introduce the problem of the evaluator knowing what a user is doing. As not all VEs present the environment on a traditional monitor but on other input/output (I/O)

devices, such as heads up displays, the evaluator cannot necessarily see what the user is experiencing just by watching. These characteristics, and others, while missing in the evaluation of GUIs, are important aspects in determining the usability of VEs [4].

There has been research done to study how physical limitations and evaluator issues effect the evaluation of usability. [4] analyzed two previously developed approaches for evaluating VEs that incorporate these usability issues. The first method is *Testbed Evaluation* which was developed by [2]. The second technique is *Sequential Evaluation* is the second technique which was developed by [5].

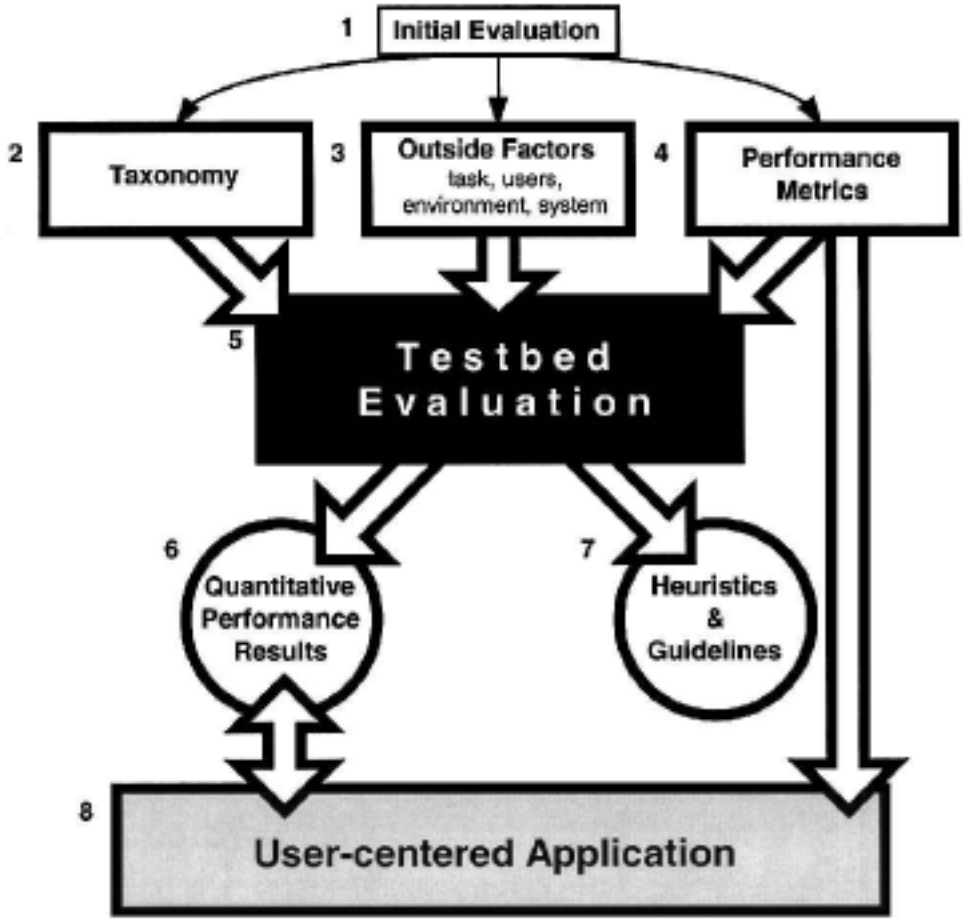


Figure 2.2: Evaluation Approach [4]

The Testbed Evaluation approach can be applied to both GUIs and VEs but is best suited for VEs due to Testbed Evaluation's focus on the low-level interactions which are inherent in VEs. Testbed Evaluation produces a set of models that characterize

a specific interaction task in terms of several metrics which relate to outside factors, such as distance traveled, number of obstacles, and even the lighting model used. In this manner, these outside factors can be treated as secondary independent variables. The results of each interaction task can be added to a database. When all tasks have completed, the database will contain all of the results from each. To evaluate a particular technique a database query can be done to find all of the tasks that make up that technique. Through analysis the results of this query can be used to determine the usability. The usability of two techniques can be easily compared using the results of their analysis [4].

The second approach to evaluating usability, Sequential Evaluation, uses a simple principal from software development and extends it to fit to VEs. Sequential evaluation has four phases: *User Task Analysis*, *Heuristic Evaluation*, *Formative Evaluation*, and *Summative Evaluation*. The first three stages are iterated over as many times as needed. Iterating over these phases provides both qualitative and quantitative results about the usability. Individual stages can be removed if it is decided that they are unnecessary. The final phase, Summative Evaluation, brings all the results of the previous phases together. In the last phase the designers can examine these results and find what parts of the VE could use improvements and even suggest ways of performing a task better [4].

Both of these approaches focus on improving the usability of VEs. The goals of each method are different, as is the underlying algorithmic approach. Testbed evaluation generates generic performance characteristics which allow the results to be applied to a wide range of techniques. The sequential evaluation improves the usability of a particular VE by finding and fixing problems during each iteration of the evaluation. It then uses the new version during the next iteration. Each of these methods is successful at evaluating usability. Only the Sequential Evaluation leads to performance enhancements as the Testbed Evaluation simply determines which technique is best [4].

To evaluate the usability of the application a testbed evaluation would provide the best information. This is because the tour is already built and the interest is in finding

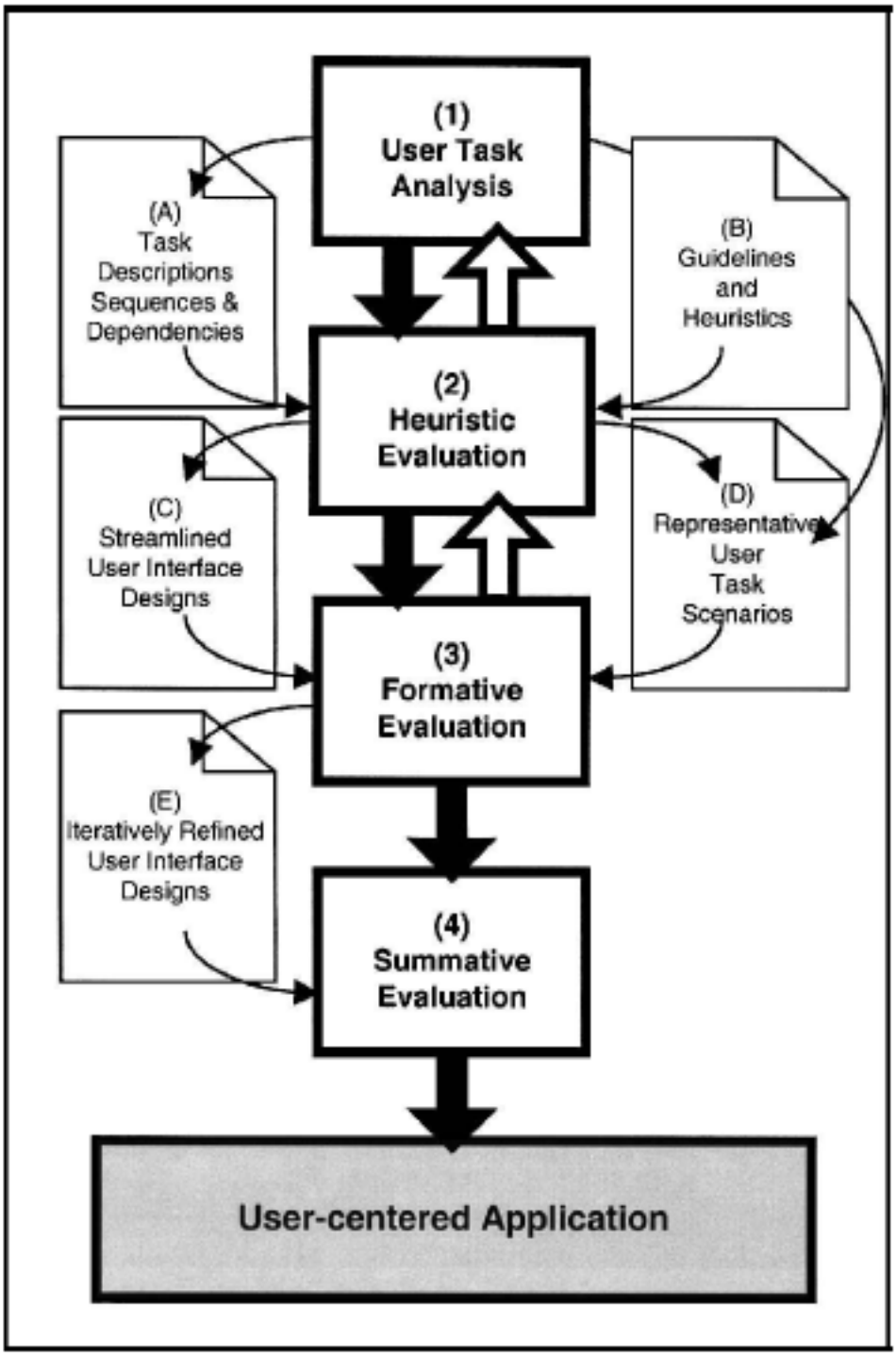


Figure 2.3: Sequential Evaluation Approach [4]

the usability of the pieces already in place. Evaluating with the testbed method would require an analysis of user performance and outside factors such as the tasks asked. This analysis combined with a taxonomy would produce performance results to make a tour more user-centered.

3 METHODOLOGY

In order to make a virtual tour we need to first select an engine for the tour while taking digital pictures of the campus in order to layout how the tour will look and feel. After this is done we build the game and perform a study on the learning performance of the tour.

3.1 Engine Selection

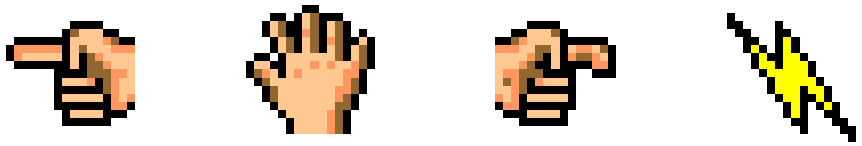
To begin, we needed an engine to form the backbone of our game. We could not make proper decisions about other aspects of the game without first selecting an engine.

3.1.1 Requirements

First, we needed to define a set of requirements for the engine. This task was fairly simple, since we had already chosen to model our game on the Myst series of games designed by Cyan Worlds, Inc.

The core game mechanic for Myst is simple. The player is shown a still image of a world from a first-person perspective. The player can then use the mouse to move a cursor around the screen. The cursor will change when it passes over an object the player can interact with. The changed cursor provides the player with an indication of what would happen upon clicking the mouse. Figure 3.1 on the next page shows some examples of cursors used in Myst. Myst also provided a “zip” command, where the player can click on a location in the distance and be instantly teleported there.

Myst also provides other sensory stimuli. When the player performs an action on a switch, the reward is a clicking noise indicating if a puzzle is solved. Ambient background music is also played to heighten the effect. Finally, video support is used to judiciously provide interaction with other characters; these interactions, however, are quite rare.



(a) Left (b) Action (c) 180° right turn (d) Zip

Figure 3.1: Cursor Examples

Our requirements, however, were somewhat more limited. We did not want to broaden our scope beyond our time constraints, so we chose to limit ourselves to still visuals only, and not include movie or sound support. We also debated over the aforementioned zip support. We decided to forego zipping because we felt that it served to remind the player of outside concerns, damaging the immersion. Our world was not large enough to justify zipping.

We also had a few requirements for the engine that did not relate to in-game activities. The engine needed to be packageable, that is, there was a way to combine all the necessary game files into a single bundle that could be run on a player's computer. That meant no library dependences unless we provided them ourselves or could rely on them to already be installed. We did not wish to lose potential players due to installation problems. We also wished to provide packages for Windows as well as for Linux.

3.1.2 Possible Engine Choices

Once we decided on our requirements, we proceeded with choosing an engine.

3.1.2.1 Pyzzle

The first option we came upon was Pyzzle, a Python-based engine. "With Pyzzle [we could] build platform independent slide show games that use the simplicity and power of Python combined with the flexibility and coolness of free (as in freedom) software[7]." The Pyzzle project's aim was to support every feature that Riven (the sequel to Myst) supported. This included the set of Myst features, as well as a few other features. It also aimed to support Riven's ability to change the color of the cursor due to game stim-

uli, and the ability to change the method by which one slide transitioned to another. It appeared to completely satisfy our requirements.

Although Pyzzle appeared on paper to meet the requirements, use of the engine proved that there were a few problems with it. We first noticed that the input format for describing a game was Python itself. This required that anyone designing the game would also be required to know Python. A better design would involve using a simple and concise custom language for designing a game. That way, a designer should be able to easily and quickly learn how to describe a game, rather than having a whole new programming language to learn. Please see Section 3.3 for more information on the solution to this problem.

With this problem solved, however, we noticed much more critical errors. Pyzzle supported defined navigation types that could be attached to slides, i.e., if a slide supported left, right, and forward navigation, it could be given the "standard" navigation type. Using the defined navigation types resulted in a code size reduction of more than half, so we tried to use them. However, we quickly found that once a navigation type was attached to a slide, the slide could not have its actions modified further. Adding a custom direction or action (a *hotspot*, in Pyzzle's terminology) would be completely and silently ignored. Therefore, for the sake of consistency, we chose to forgo the navigation types and instead use custom hotspots. However, with this change, we noticed that many slides were no longer behaving correctly. Slides with properly-defined right turns would not allow these right turns once the game was running. We attempted to track this problem down. In doing so, we discovered that the Python code was quite poorly written and lacked comments. The code was littered with duplicate code and unused functionality. We found it quite difficult to navigate through the code. We finally narrowed the problem down to the custom types we had chosen to use. The reason that right turns were universally the problem was that right turns were always added to a slide last, after the left and straight-ahead navigations. Pyzzle refused to add more than two custom hotspots to any given slide and without any indication of error. Pyzzle's maintainer had declared Pyzzle abandoned in November of 2004. We knew that we could not expect aid from him when we had problems with Pyzzle. We

began to look for alternate solutions, and found Il—— [1].

3.1.2.2 Il——

We began searching the abandoned site for clues about what we could do next. We found a small post on Pyzzle's forum about a group creating a fork of Pyzzle to continue development. This group called their engine Il——. They had released a demo of their product, so we downloaded it and tried it out. It was clearly a fork of Pyzzle containing about half of the files Pyzzle had, but what was there was the same. The demo, however, was distributed as compiled Python bytecode. Python bytecode is directly tied to the major and minor version numbers of the python executable it is compiled with. We attempted to disassemble the bytecode, but only a few files could be decompiled. We knew if we were going to have any chance of distributing a python-based engine in a cross-platform manner, we were going to need the source. So we wrote an email to them requesting a copy of the source. We assumed that, if they were assembling an actual product with their engine, they would have fixed the bugs that plagued Pyzzle. We were very polite, and offered to publicize them as much as we could in return. We then spent the next few weeks waiting. One member of the team told us who to contact to get the source, but that contact never responded. Finally, as we began running into deadlines, we sent a final request. An hour after the final deadline we set, the silent contact finally responded, providing a compiled-bytecode version of the engine instead of the source we needed.

At this point, we decided to forego Il—— and resume using Pyzzle, bugs notwithstanding. We began the packaging process, and discovered another bug that caused us even more difficulty. Python distributes a script called "Freeze", which compiles a python script into a stand-alone executable suitable for distribution to any like-architectured computer. We were counting on this script for our final release. However, as we began to use it, we discovered that it did not work in all cases. Specifically, it could not Freeze any python script that depended on a python library implemented in

[1] We have redacted the name of this engine. As described in Section 3.1.2.2, they were utterly unhelpful. In addition, we believe them to be in violation of the GNU General Public License. As such, we do not wish to provide them with any publicity, and have omitted their name.

a language other than python. Pyzzle depended on Pygame, a library implemented in C. This limitation confused us since we had the source to Pygame; why should Freeze have so much trouble with this? This was a show-stopping problem because we had to be able to distribute our project. We decided that we could either continue fighting with Freeze, or we could write our own engine in C, which we already knew how to package in a cross-platform manner. We chose the latter.

3.1.2.3 Custom Engine

With the decision made, we knew we had to write it as quickly as possible. We still needed it to be cross-platform, so we decided to use as few libraries as possible. We also stuck to libraries we already knew were completely cross-platform. We decided that we were going to keep all of the rest of our support code the same. Namely, we would store the game in the XML format we had already decided precisely for this purpose. Therefore, we needed an XML library. The GNOME project had a cross-platform XML library, but it was quite complicated to pick up quickly. However, the Gaim Instant Messenger client contained a quite small, very simple XML library `xmlnode.c` (Listing A.5 on page 41). Since this project was Open Source, we decided to use this library. After a few hours of rapid coding, we came up with `main.c` (Listing A.2 on page 30), the engine. It is a reimplementation of many of Pyzzle's features, but does not share code or design with Pyzzle.

3.2 Making Slides

To properly display information about the campus some form of image was needed. Modeling the environment in a program such as Autodesk's 3D Studio Max would provide a high quality image that the user could easily find information about their surroundings. This however requires a large amount of time to model the exterior of the buildings and the pathways that connects them. Another medium is digital pictures which has the possibility of producing a lower quality image but does not require modeling. The reduction in image quality is not significant enough that the

user would not be able to find information about the campus.

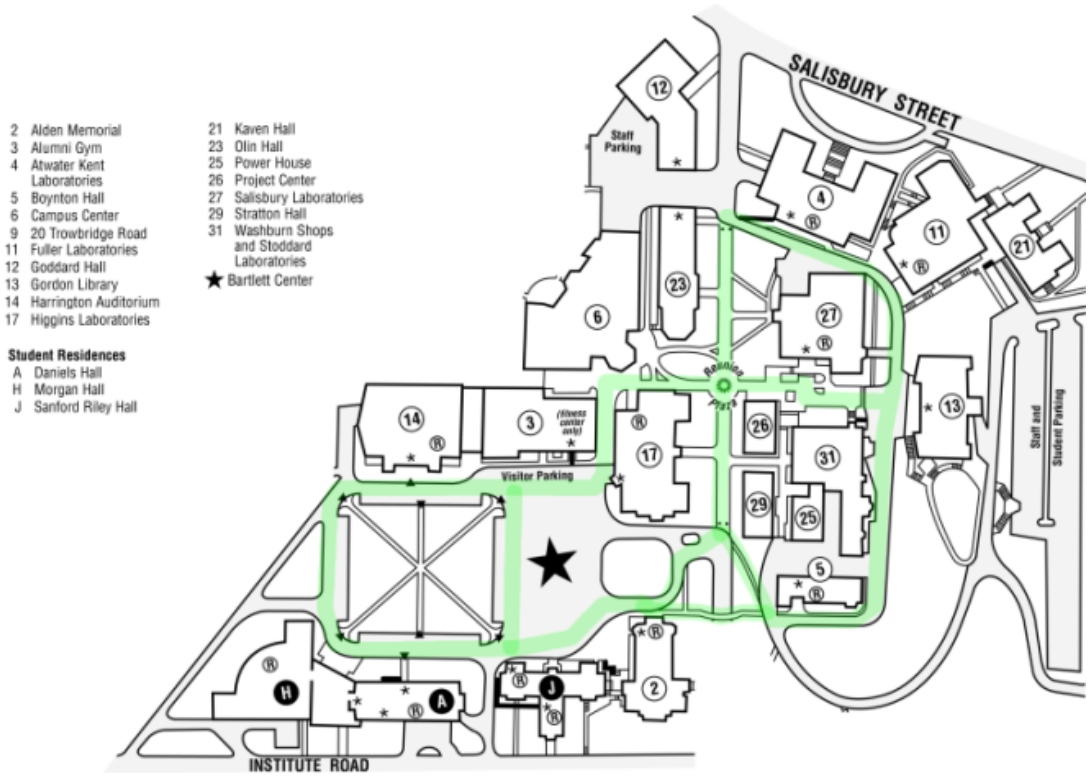


Figure 3.2: The region of WPI covered by our game

Choosing between these mediums was not difficult due to the limited time to complete the project. The biggest challenge of using digital pictures was finding a suitable distance between each picture. Another way of stating this problem is how many steps should be taken between each shot. To find the number of steps needed several pilot tests were done using step sizes of 14, 12, 10, 8 and 6. Flipping through each set of images quickly produces a sense of traveling. By comparing how realistic this sense is we found the proper distance. It was determined that a step size of 8 produced the closest match to walking at a steady pace. Setting the step size to a constant number had the effect of limiting the number of pictures required to cover all major outdoor portions of campus outlined by Figure 3.2. After taking all the pictures the limit was found to be between 450 and 500 pictures.

Another concern was the disk space needed for all the images. In order to ensure an easy way to distribute the program, a maximum disk usage had to be decided upon.

The easiest way of distributing would be to have user download a compressed version from the internet. This means that the maximum size had to be under 50 MB so slow download speeds would not be a deterrent to using the tour.

Using a 3.1 mega-pixel camera on landscape and highest quality produced a 2160px by 1440px image that approximately requires 1.5 MB of disk space. At this size the total space to hold the upper bound of 500 pictures would be 750 MB, clearly too large to easily distribute. One method to decrease the space needed is to shrink the pictures to a small size. Choosing a size that fits on an average monitor was easy since there are only a few default screen sizes. The smallest screen size of 800px by 600px makes a good choice for all monitors. This change decreased the size of one image to approximately 70 KB, making a total disk usage of 35 MB. This improvement clearly is much better and can easily be downloaded in a reasonable time.

3.3 Building Game

To build a working game, we created an assembly-line of tools to take in a set of connected slides and output the finished product. It was not necessary to change any of these tools when we switched between different engines; we designed them to be useful under any circumstance. The only exception was xml2py.pl (Section 3.3.2) which became unnecessary when we wrote our custom engine; this engine's input format was the format described by `game.dtd` (Listing A.7 on page 53), and no additional conversion was necessary.

3.3.1 game.dtd

We decided that our first task was to create a description language to describe the game. The natural choice for this was XML, since there are a multitude of tools designed to manipulate it. We first decided on our requirements and wrote a draft of `game.dtd` (Listing A.7 on page 53). This first version suited our requirements for the Pyzzle engine. When we decided to switch engines, we reused the same format. It worked exactly as we had designed it— it could describe a game independent of the

engine. Unfortunately, our custom engine does not fully support the DTD— a detail that can hopefully be rectified by a later project.

3.3.2 xml2py.pl

Next, we needed to convert our XML format into Python using Pyzzle's API. This script, `xml2py.pl` (Listing A.10 on page 57), consumed a game in XML, checked it for obvious errors, and output a game that Pyzzle could read. The first version output custom hotspots for each direction. As detailed above, however, Pyzzle could not handle more than two custom hotspots. To mitigate this problem, we added a simple heuristic to the script to pick a defined navigation type instead of using custom hotspots. This solution would not work in the general case, but we did not have any slides that triggered the failure case.

3.3.3 path-to-xml.pl

We also needed some way to enter the path data. The path data is a directed graph that describes the interconnections between the slides. We decided to use an Excel spreadsheet as a medium for this. We wrote a small script to consume a number of Excel spreadsheets, do some basic error-checking, and output an XML file in the game.dtd format.

3.3.4 path-to-graph.pl

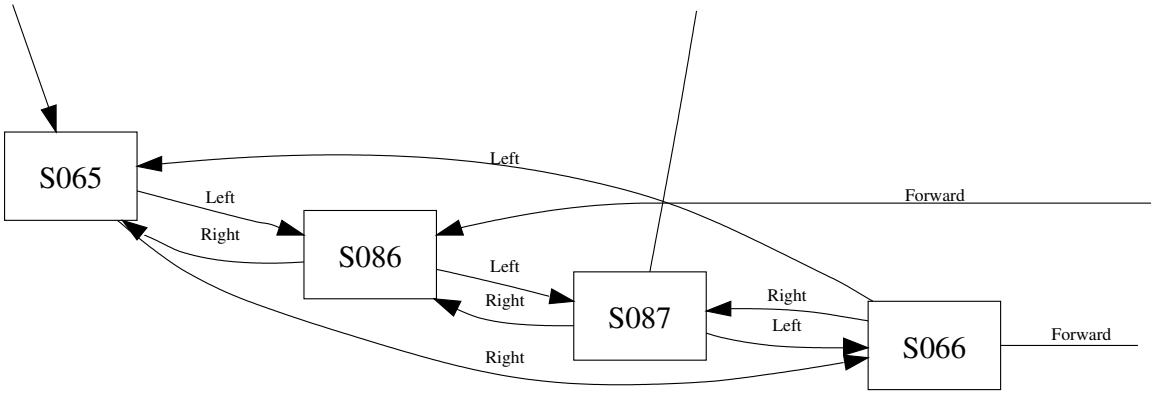


Figure 3.3: Example graph created with path-to-graph.pl

Finally, we needed as many ways to catch and debug errors as possible. This script, modelled off of the previous script, created a directed graph of the slides. Some classes of errors made themselves quite obvious in this format. For instance, by visual inspection of this graph it was quite easy to see a case where turning left and then immediately right again would not result in the same view. It would appear on the graph as a node with a path to second node that did not have a direct path back to the original node.

3.4 User Survey

To determine if the tour would help people to learn about WPI a user study is required. Keeping the goal of helping students to find and travel between places in mind, a survey would be the appropriate method to study users. In order for the survey to be effective there had to be two parts to it, one before playing the game and another after. The questions, shown in Appendix C on page 72, for the two parts needed to be the same in order to allow a straight-forward comparison of responses.

Collecting demographics is important to find and eliminate outliers such as current or past students. By asking people if they are a student we can eliminate them from the analysis of the responses. Another possible problem would be people who have previous knowledge of campus. This can be handled by asking the user if they have visited the campus or have been part of a tour. To see the questions used see Appendix B on page 71.

A guideline of survey design is to keep the number of questions to a small amount in order to maximize response and accuracy. Using this guideline, ten questions were decided on because it should take five minutes to complete this amount of questions. By breaking these questions up further to three types of questions allows for the understanding of how the user interprets the spatial relations, path selection and general knowledge. Spatial relation questions were used to test how the user perceived where different buildings were located in respect to others. This type of testing would be useful in a tour application as many people navigate by the use of landmarks instead of

Question	**Choices**
Salisbury Labs is directly located between Boynton Hall and Atwater Kent?	• True • False • Don't Know
The Birch Tree is between _____ and _____.	• Atwater Kent, Goddard Labs • Higgins Labs, Alden Memorial • Morgan Hall, Daniels Hall • Don't Know
Alumni Gym is next to Stratton Hall?	• True • False • Don't Know
Daniels Hall is connected to _____.	• Morgan Hall • Riley Hall • Olin Hall • Don't know

Table 3.1: Spatial relation questions and answer choices

path directions.

The second type of question was path selection that asked the user to find the path that would get the player from one particular place to another. The difficulty with these questions was denoting what the direction of travel was and how the user was initially facing. Since there was no compass or key to north this eliminated using compass points to show travel and facing. Using places that had only one initial facing eased the problem of telling the user how they start. It also helped to determine how to update the user if the direction of travel changed. By using words such as left, right or straight to indicate how the user was traveling the tour could follow the natural steering mechanism.

General knowledge questions were used to study if the user could learn about different the departments and facilities. This type of knowledge would be difficult for the user to gain without having some form of interaction with the inside of buildings;

Question	**Choices**
From the start, how would you get from Reunion Plaza to the campus center?	• Turn right and follow the path to the end and then turn right • Go straight until the path ends, turn left and go straight until the end of the path • Turn left and enter the Campus center • Don't Know
How would you get from Morgan Hall to Harrington Auditorium?	• Turn left and follow the path until it goes into Harrington • Follow the path straight, after the turn in the path turn left • Turn right, follow the path and then turn left • Don't Know
How would you get from Olin Hall to Salisbury Labs?	• Go straight until you enter Salisbury • Turn left, follow path to the end turn right and follow path until the end • Turn around and enter building • Don't Know

Table 3.2: Path selection questions and answer choices

Question	**Choices**
Inside of Fuller Labs you can find the _____ Dept.	• Electrical and Computer Engineering • Biology • Computer Science • Don't Know
The Music Dept. is in _____.	• Bartlet Center • Riley Hall • Alden Memorial • Don't Know
Boynton Hall currently holds the administration?	• True • False • Don't Know

Table 3.3: General knowledge questions and answer choices

however this type of knowledge would be useful for a student looking to know more about WPI. Studying how the tour helps in this area gives an indication of where improvement could be made if additional information was to be imparted.

The study was distributed to five high school teachers using an e-mail. The e-mail had information about the project and asked the teachers to give the attached flier to their students. The flier had a better description of the project and included the web address of the survey. A few teachers gave the e-mail and flier to their colleagues to give to students. One setback with our approach was that one school the e-mail was sent to gave each of its students an Apple and these students could not run the tour, since we lacked an Apple build.

4 Analysis

To determine if the tour helped those who completed the study a analysis of the results had to be done. We first looked at the demographic data to see if there were any responses that could be eliminated and prevent corrupting the results. Next we analyzed the results by looking at the difference in responses between the two surveys.

4.1 Demographics

Fifty one people started to take the survey but only thirty people completed the post survey. To properly evaluate the response that was received it was necessary to remove the responses from people who did not complete the survey. This avoids skewing the data in a way that would give a false conclusion due to the greater number of responses for the pre survey.

We broke down the thirty people into four age groups. The first age group of under sixteen (underclassmen) had four people. The next age group of between sixteen and eighteen (upperclassmen) had eight people. These groups combined make up one third of the total responses and is the intended age of the audience for the tour since the tour is meant to help them learn more the colleges they might apply to. The next two thirds contain two people between nineteen and twenty two and sixteen people over the age of twenty two. This age breakdown allows for refining the data to ensure that any conclusions made from the data hold for the representatives of the sixteen to eighteen target group.

It is also necessary to eliminate the possibility of corrupted data from those who have extensive knowledge of the campus. This category would include current or past students and those who have participated in tours. Answering yes to being a student or answering yes to both having visited and taken a tour would indicate a possibility of knowing too much. Out of the thirty users there were no students and seven people

to visit campus, three of which were part of a tour.

4.2 Results

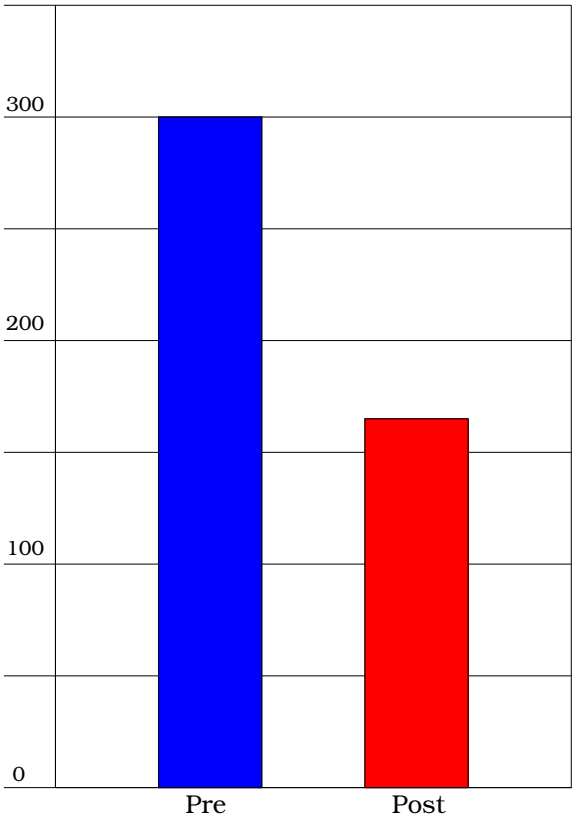


Figure 4.1: “Don’t Know” responses

Grouping the questions into their three categories, spatial relation, path selection and general knowledge, produces an image of how users performed in each area of interest. Figure 4.1 shows the comparison of “Don’t Know” responses in the pre survey to the post survey and shows there was a decrease of forty five percent. Because no one answered differently then Don’t Know we compared the number of “Don’t Know” answers before and after using the tour. It became evident that users felt more comfortable answering the questions after using the tour.

Examining how the responses were distributed amongst the three areas would provide an image of any improvement. Taking a closer look, eighty two percent of the responses were correct. This indicates that the users were able to take something

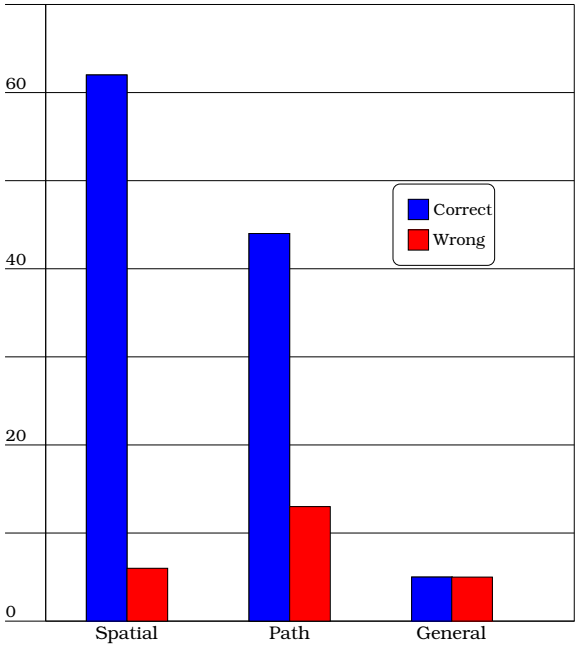


Figure 4.2: Post-survey answer breakdown

away from using the tour. Figure 4.2 shows the distribution between the three areas showing that users could tell the spatial relationships between buildings.

A person who guessed at the answers would have a one in three chance to select the correct answer. This means that on average a person who guessed would have three correct responses. There was two division in the responses, one where people had no correct answers and one where the average number of correct responses was six. The first division had eleven people while there was nineteen people in the other. There were a few people who scores threes and fours and possibly guessed but both divisions showed that the majority of responses were not guesses.

The reason there was no improvement on the general knowledge questions is because the user did not have access to the inside of buildings. Without seeing the inside of the buildings a user would not be able to answer questions about the contents of the building. There was also no content to help the user learn about parts of campus just looking at an image can teach.

5 Conclusion

People want to remotely learn about a physical space to avoid the cost of travel or because they can't travel to the physical location. Virtual tours bring a physical space to remote users and provide an interactive view. Hotels and universities such as Worcester Polytechnic Institute (WPI) uses this tool to give patrons or prospective students insight. Virtual tours can be made using virtual environments, videos with limited interactivity or VRML views.

Building a virtual tour system requires displayable media and some method of showing the media. By taking pictures of campus and displaying them using a custom engine we produced a virtual tour of walking on the WPI campus. To evaluate the effectiveness of the tour's teaching ability a user study had to be undertaken. Analyzing the results of the study by comparing the change in responses before and after using the application provides an idea of the user comfort answering questions. Also looking at the distribution between correct and wrong answer gives insight to any improvement the use may have made.

After completing the user survey we found that users had increased knowledge of relationship between buildings and how to choose a path between them. The findings also found that the tour did not help users to learn general information about the university. This is due to the lack of content required to impart this type of information.

Our contributions include a set of pictures of the campus along with an engine to move in the virtual environment. This set of pictures and the engine can be expanded to provide a better look at WPI as laid out in Chapter 6, Future Work. We also created a framework for completing a user study. Comparing the results of this study to another would provide insight into the degree of improvement made by furthering the project.

6 Future Work

Testing showed that people who used the application would be able to walk from a portion of campus to another and have a general idea of where they are. This gives the impression that with future work the application would provide a true perspective of campus and life at WPI. To attain this, the tour needs to include indoor pictures, interaction content and entertainment content.

Allowing the user to go inside of the buildings to see labs and classrooms is a good way to increase general knowledge and also generate a greater interest in the school. People who have an interest in a particular major would be able to see the facilities such as labs and equipment open to students of that major. To do this more slides would need to be added to the tour. This encompasses taking the new pictures, running them through the reduction process described in the methodology and adding the new paths definitions. The difficulty of this would be finding an easy way to travel from floor to floor in each building. Stairwells did not provide enough space for the standard step size and required to be able to break the direction selection into two or more pieces.

Interaction and entertainment content play a similar role in the fact that they both add an interest level. This provides the user with something to do or teaches the user about something. Interaction content would be activities or tasks a normal first year student has to do on arriving at WPI such as picking up their ID and residence hall keys. This would help to reduce the confusion and frustration felt by first years because they know what is expected of them and how to do it.

The entertainment content has little or no teaching value but provides more fun to the tour to make it closer to the game that was original intended. Examples could be finding a missing key or catching the stray dog that stole the users lunch. This type of content is meant to provide entertainment and to make the user want to keep playing. It has the added benefit of reducing the redundancy of playing because content is based

off of previous actions.

In order to implement interaction and entertainment content some sort of action scripting must be added to the custom engine. Creating a simple language that defines how to handle certain events such as picking an object up, tracking inventory and quests, and using an item would produce an infinite amount of activities that a user could be asked to do. There would also need to be some way to define a clickable part of the slide that would perform the desired action. The semblance of this is already built into the engine with the ability to turn left and right but a more sophisticated system to determine what action to take would need to added to ensure that the user did not trigger the wrong event.

BIBLIOGRAPHY

[1] Begun and Stroup. Campus tours 1.0. *Newsweek*, 136(18):97, 2000.

[2] Bowman and Hodges. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *The Journal of Visual Languages and Computing*, 10(1):37–53, 1999.

[3] Bowman, Koller, and Hodges. A methodology for the evaluation of travel techniques for immersive virtual environments. *Virtual Reality*, 3(2):120–131, 1998.

[4] Bowman, Gabbard, and Hix. A survey of usability evaluation in virtual environment: Classification and comparison of methods. *Presence*, 11(4):404–424, 2002.

[5] Gabbard, Hix, and Swan. User centered design and evaluation of virtual environments. *IEEE Computer Graphics and Applications*, 19(6):51–59, 1999.

[6] Il—— Team. *The Ages of Il—— Web Site*, 2006. More information available upon request.

[7] Andrew Jones. *P Y Z Z L E: The Cross Platform Open Source Myst/Riven Interface Engine*, 2004. URL: <http://pyzzle.sourceforge.net/>.

APPENDIX A: CODE

Sample XML data file describing a game

Listing A.1: `data.xml`

```
<game>
    <slide>
        <file>S001</file>
        <next>
            <forward>S002</forward>
            <left>S057</left>
            <right>S094</right>
        </next>
    </slide>
    <slide>
        <file>S002</file>
        <next>
            <forward>S003</forward>
            <left180>S029</left180>
            <right180>S029</right180>
        </next>
    </slide>
    <slide>
        <file>S003</file>
        <next>
            <forward>S004</forward>
            <left180>S028</left180>
            <right180>S028</right180>
        </next>
    </slide>
    <slide>
        <file>S004</file>
        <next>
            <forward>S005</forward>
            <left180>S027</left180>
            <right180>S027</right180>
        </next>
    </slide>
    <slide>
        <file>S005</file>
        <next>
            <forward>S006</forward>
            <left180>S026</left180>
            <right180>S026</right180>
        </next>
    </slide>
</game>
```

Main engine code

Listing A.2: `main.c`

```
#include <stdio.h>

#include <glib.h>
#include <SDL.h>
#include <SDL_image.h>
#include <string.h>

#include "xmlnode.h"
#include "main.h"

#define SWID 800
#define SHGT 533
#define WWID 912
#define WHGT 645
#define OFFX ((WWID-SWID)/2)
#define OFFY ((WHGT-SHGT)/2)
#define EDGE 120
#define ARRX 25
#define ARRY 25

static gboolean debug = FALSE;

typedef struct {
    gchar* id;
    gchar* left;
    gchar* right;
    gchar* forward;
    gchar* left180;
    gchar* right180;
    gchar* up;
    gchar* down;
    gchar* fn;
} Slide;

typedef enum {
    R_MIDDLE,
    R_LEFT,
    R_RIGHT,
    R_TOP,
    R_BOTTOM,
    R_OFF,
    R_INVALID
} Region;

typedef enum {
    C_ARROW,
    C_LEFT,
    C_LEFTSPIN,
    C_RIGHT,
    C_RIGHTSPIN,
    C_UP,
    C_DOWN,
    C_FWD,
    C_COUNT
} Cursor;

SDL_Surface *curs[C_COUNT];
gchar *curfiles[C_COUNT] = {"arrow.png", "left.png", "left180.png", "right.png",
                            "right180.png", "up.png", "down.png", "fwd.png"};

SDL_Surface *arrows[C_COUNT];
gchar *arrowfiles[C_COUNT] = {NULL, "arrow-left.png", "arrow-left180.png", "arrow-right.
    png",
```

```
                                "arrow-right180.png", "arrow-up.png", "arrow-down.png", "
                                    arrow-fwd.png"};

SDL_Rect arrowrect[C_COUNT] = {{0,0,0,0},
                              {(OFFX-ARRX)/2, (WHGT-ARRY)/2, 0, 0}, {(OFFX-ARRX)/2, (WHGT-
                                  ARRY)/2, 0, 0},
                              {WWID-((OFFX+ARRX)/2), (WHGT-ARRY)/2, 0, 0}, {WWID-((OFFX+ARRX
                                  )/2), (WHGT-ARRY)/2, 0, 0},
                              {WWID/2, EDGE/2, 0, 0}, {WWID/2, SHGT+(EDGE/2), 0, 0},
                              {(WWID-ARRX)/2,(OFFY-ARRY)/2,0,0}};

Region get_region(int x, int y)
{
    x -= OFFX;
    y -= OFFY;

    if (x < 0 || x > SWID)
        return R_OFF;
    if (y < 0 || y > SHGT)
        return R_OFF;

    if (x < EDGE)
        return R_LEFT;
    if (x > SWID-EDGE)
        return R_RIGHT;

    if (y < EDGE)
        return R_TOP;
    if (y > SHGT-EDGE)
        return R_BOTTOM;

    return R_MIDDLE;
}

void slide_free(Slide *s)
{
    g_free(s->id);
    g_free(s->left);
    g_free(s->right);
    g_free(s->forward);
    g_free(s->left180);
    g_free(s->right180);
    g_free(s->up);
    g_free(s->down);
    g_free(s->fn);
    g_free(s);
}

inline gchar *cdata(xmlnode *p, gchar *cname)
{
    xmlnode *x = xmlnode_get_child(p, cname);
    if (x)
        return xmlnode_get_data(x);
    else
        return NULL;
}

void view(gchar *key, Slide *s, gpointer null)
{
    if (debug)
    {
        printf("Slide (id=%s, fn=%s): ", s->id, s->fn);
        #define printif(str) do {\
            if (s->str){printf("(%s %s) ", #str, s->str);}\
        } while (0)
        printif(forward);
        printif(left);
        printif(right);
```

```
        printif(left180);
        printif(right180);
        printif(up);
        printif(down);
        printf("\n");
    }
}

GHashTable* make_weak()
{
    gchar *data;
    gint len;
    GError *err = NULL;
    xmlnode *top,*s;
    GHashTable *slides;
    gchar *fn,*path;

    if (!g_file_get_contents("data.xml", &data, &len, &err))
    {
        warn("Couldn't open %s: %s\n", "data.xml", err->message);
        g_error_free(err);
        die("Failed.");
    }

    top = xmlnode_from_str(data, len);
    g_free(data);

    s = xmlnode_get_child(top, "slide");

    slides = g_hash_table_new_full(g_str_hash, g_str_equal, g_free, (GDestroyNotify)
        slide_free);

    while (s)
    {
        gchar *id = cdata(s, "file");
        if (g_hash_table_lookup(slides, id))
        {
            warn("Slide %s was already seen. Ignoring the new one.\n", id);
            g_free(id);
        }
        else
        {
            Slide* tmp = g_new0(Slide, 1);
            xmlnode *next = xmlnode_get_child(s, "next");
            tmp->id = g_strdup(id);
            if (next)
            {
                #define try_insert(slide, node, child) do {\
                    gchar *d = cdata(node, #child);\
                    if (d) {\
                        slide->child = d;\
                        fn = g_strconcat(tmp->id, ".jpg", NULL);\
                        path = g_build_filename(SLIDEDIR, fn);\
                        if (!g_file_test(path, G_FILE_TEST_EXISTS))\
                            warn("%s does not exist at %s\n", fn, path);\
                        g_free(fn);\
                        slide->fn = path;\
                    }\
                } while (0)
                try_insert(tmp, next, left);
                try_insert(tmp, next, right);
                try_insert(tmp, next, forward);
                try_insert(tmp, next, left180);
                try_insert(tmp, next, right180);
                try_insert(tmp, next, up);
                try_insert(tmp, next, down);
            }
            g_hash_table_insert(slides, id, tmp);
```

```
        }

        s = xmlnode_get_next_twin(s);
    }
    xmlnode_free(top);
    Slide* tmp = g_new0(Slide, 1);
    tmp->id = g_strdup("map");
    fn = g_strconcat(tmp->id, ".jpg", NULL);
    path = g_build_filename(SLIDEDIR, fn);
    if (!g_file_test(path, G_FILE_TEST_EXISTS))
        warn("%s does not exist at %s\n", fn, path);
    g_free(fn);
    tmp->fn = path;
    g_hash_table_insert(slides, g_strdup(tmp->id), tmp);

    return slides;
}

void draw_slide(SDL_Surface *surface, gchar *fn)
{
    static SDL_Surface *image = NULL;
    static gchar *last = NULL;

    if (last != fn)
    {
        if (image != NULL)
            SDL_FreeSurface(image);
        if (!(image = IMG_Load(fn)))
            die("IMG_Load: %s\n", IMG_GetError());
        last = fn;
    }

    SDL_Rect rcDest = { OFFX, OFFY, 0, 0 };
    SDL_BlitSurface(image, NULL, surface, &rcDest);
}


SDL_Surface* load(const gchar *path)
{
    SDL_Surface *image;
    if (!(image = IMG_Load(path)))
        die("IMG_Load: %s\n", IMG_GetError());
    return image;
}

void init_cursors()
{
    gchar *fn;
    int i;
    for (i = 0; i < C_COUNT; ++i)
    {
        fn = g_build_filename(CURSORDIR, curfiles[i], NULL);
        curs[i] = load(fn);
        g_free(fn);

        if (arrowfiles[i])
        {
            fn = g_build_filename(CURSORDIR, arrowfiles[i], NULL);
            arrows[i] = load(fn);
            g_free(fn);
        }
    }
}

void update_cursor(SDL_Surface *s, Slide *current)
{
    SDL_Surface *image;
    Region next;
```

```
    int x,y;

    SDL_GetMouseState(&x, &y);

    next = get_region(x,y);

    image = curs[C_ARROW];
    switch (next)
    {
        case R_LEFT:
            if (current->left180)
                image = curs[C_LEFTSPIN];
            else if (current->left)
                image = curs[C_LEFT];
            break;

        case R_RIGHT:
            if (current->right180)
                image = curs[C_RIGHTSPIN];
            else if (current->right)
                image = curs[C_RIGHT];
            break;

        case R_TOP:
            if (current->up)
                image = curs[C_UP];
            break;

        case R_BOTTOM:
            if (current->down)
                image = curs[C_DOWN];
            break;

        case R_MIDDLE:
            if (current->forward)
                image = curs[C_FWD];
            break;

        default:
            break;
    }

    SDL_Rect rcDest = {x, y, 0, 0};
    SDL_BlitSurface(image, NULL, s, &rcDest);
}

void draw_arrow(SDL_Surface *s, Cursor id)
{
    SDL_Rect *rcDest = &arrowrect[id];
    if (arrows[id])
        SDL_BlitSurface(arrows[id], NULL, s, rcDest);
    else
        SDL_BlitSurface(curs[id], NULL, s, rcDest);
}

void draw_exits(SDL_Surface *s, Slide *current)
{
    if (current->left180)
        draw_arrow(s, C_LEFTSPIN);
    else if (current->left)
        draw_arrow(s, C_LEFT);

    if (current->right180)
        draw_arrow(s, C_RIGHTSPIN);
    else if (current->right)
        draw_arrow(s, C_RIGHT);

    if (current->up)
```

```
        draw_arrow(s, C_UP);

    if (current->down)
        draw_arrow(s, C_DOWN);

    if (current->forward)
        draw_arrow(s, C_FWD);
}

Slide *try_move(Slide* cur, Region r, GHashTable *h)
{
    gchar *id = NULL;
    switch(r)
    {
        case R_LEFT:
            if (cur->left180)   id = cur->left180;
            else if (cur->left) id = cur->left;
            break;

        case R_RIGHT:
            if (cur->right180)   id = cur->right180;
            else if (cur->right) id = cur->right;
            break;

        case R_TOP:
            id = cur->up;
            break;

        case R_BOTTOM:
            id = cur->down;
            break;

        case R_MIDDLE:
            id = cur->forward;
            break;

        default:
            break;
    }
    if (id)
    {
        if (debug)
            warn("%s -> %s\n", cur->id, id);
        cur = g_hash_table_lookup(h, id);
        view(NULL, cur, NULL);
    }
    return cur;
}

int main(int argc, char **argv)
{
    SDL_Surface *screen;
    SDL_Event event;
    Slide *cur, *last = NULL;
    GHashTable *h = make_weak();
    gboolean update = 1, draw_mouse = 1;

    if (SDL_Init(SDL_INIT_VIDEO) < 0)
        die("Unable to init SDL: %s", SDL_GetError());

    if (!(screen = SDL_SetVideoMode(WWID, WHGT, 0, SDL_SWSURFACE)))
        die("Could not get a surface: %s", SDL_GetError());

    SDL_ShowCursor(SDL_DISABLE);
    init_cursors();

    cur = g_hash_table_lookup(h, "S001");
    view(NULL, cur, NULL);
```

```
    while(1)
    {
400     while(SDL_PollEvent(&event))
        {
            switch(event.type)
            {
                Region r;
405             case SDL_QUIT:
                    goto bail;
                case SDL_KEYDOWN:
                    r = R_INVALID;
                    switch (event.key.keysym.sym)
410                 {
                        case 'd':
                            debug = debug ? FALSE : TRUE;
                            warn("Debug Mode %s\n", debug ? "enabled" : "disabled");
                            goto noadd;
415
                        case SDLK_TAB:
                            if (last)
                            {
                                if (debug)
420                                 warn("%s -> %s\n", cur->id, last->id);
                                cur = last;
                                last = NULL;
                                view(NULL, cur, NULL);
                            }
425                         else
                            {
                                last = cur;
                                if (debug)
                                    warn("%s -> %s\n", cur->id, "map");
430                             cur = g_hash_table_lookup(h, "map");
                                view(NULL, cur, NULL);
                            }
                            goto noadd;

435                     case 'q':
                        case SDLK_ESCAPE:
                            goto bail;
                        case SDLK_LEFT:
                            r = R_LEFT;
440                         break;
                        case SDLK_RIGHT:
                            r = R_RIGHT;
                            break;
                        case SDLK_UP:
445                         r = R_MIDDLE;
                            break;
                        /* This one breaks the plane paradigm (up is forward,
                         * so down should be backward) so it's getting commented.

450                         case SDLK_DOWN:
                            r = R_BOTTOM;
                            break;*/
                        default:
                            break;
455                 }
                    if (r != R_INVALID)
                        cur = try_move(cur,r,h);
                    noadd:
                    break;
460             case SDL_ACTIVEEVENT:
                    draw_mouse = (event.active.state&SDL_APPMOUSEFOCUS) ? event.active.
                        gain : 1;
                    update     = (event.active.state&SDL_APPACTIVE)     ? event.active.
                        gain : 1;
                    break;
```

```
                case SDL_MOUSEBUTTONDOWN:
                    if (event.button.state  == SDL_PRESSED &&
                            event.button.button == SDL_BUTTON_LEFT)
                    {
                        Region r = get_region(event.button.x, event.button.y);
                        cur = try_move(cur, r, h);
                    }

                    break;
                default:
                    break;
            }
        }
        if (update)
        {
            SDL_FillRect(screen, NULL, 0);
            draw_slide(screen, cur->fn);
            draw_exits(screen, cur);
            if (draw_mouse)
                update_cursor(screen, cur);
            SDL_UpdateRect(screen, 0, 0, 0, 0);
        }
        else
            SDL_Delay(250);
    }

bail:
    SDL_Quit();
    g_hash_table_destroy(h);
    return 0;
}

/* $Revision: 1.11 $ */
```

Main header

Listing A.3: `main.h`

```
#ifndef MAIN_H
#define MAIN_H

#include <stdlib.h>
#define SLIDEDIR "slides"
#define CURSORDIR "cursors"

#define warn(...) fprintf(stderr, __VA_ARGS__)
#define die(...) {warn(__VA_ARGS__); exit(-1);}

/* $Revision: 1.2 $ */
#endif
```

Rules to build the engine for Windows or Linux

Listing A.4: `Makefile`

```
CC=gcc
WINCC:=/opt/xmingw/bin/$(CC).exe
OBJS=main.o xmlnode.o

PKGS=glib-2.0
PKG_CONFIG=PKG_CONFIG_PATH=lin-libs/lib/pkgconfig pkg-config
CFLAGS=-Wall -W -Wno-unused-parameter\
        `$(PKG_CONFIG) --cflags $(PKGS)` -DG_DISABLE_DEPRECATED\
        `sdl-config --cflags`

WINFLAGS=-I/opt/xmingw/i386-mingw32msvc/include -Dmain=SDL_main
WINLIBS=-L/opt/xmingw/i386-mingw32msvc/bin -lmingw32 -lSDLmain -lSDL\
        -lglib-2.0-0 -lSDL_image -mwindows

LIBS=-Wl,-Bstatic -lSDL_image -lSDL -ltiff -ljpeg \
    `libpng-config --static --libs` `${PKG_CONFIG} --libs $(PKGS)` \
    -Wl,-Bdynamic -L/usr/lib -Wl,-rpath,/usr/lib -lpthread -lm -ldl -lX11 -lXext

PROG=wpi-tour

.PHONY: all clean tags split dist dist-linux dist-win32 dist-data \
        dist-linux-nodata dist-win32-nodata

all: ${PROG} ${PROG}.exe

$(PROG).exe : ${OBJS:.o=-win.o}
    $(WINCC) ${CFLAGS} $(WINFLAGS) $^ -o $@ ${WINLIBS}

$(PROG) : ${OBJS}
    $(CC) ${CFLAGS} $^ -o $@ ${LIBS}

%-win.o : %.c
    $(WINCC) ${CFLAGS} $(WINFLAGS) -c $^ -o $@

%.o : %.c
    $(CC) ${CFLAGS} -c $^ -o $@

clean:
    rm -f $(PROG) $(PROG).exe core core.* *.o

distclean: clean
    rm -fr dist

tags: *.c *.h
    exuberant-ctags *.c *.h

split: dist-linux-nodata dist-win32-nodata dist-data

dist: dist-linux dist-win32

dist-linux: wpi-tour
    mkdir -p dist/wpi-tour
    cd dist &&\
        ln -s ../../cursors wpi-tour &&\
        ln -s ../../data.xml wpi-tour &&\
        ln -s ../../slides wpi-tour &&\
        ln -s ../../wpi-tour wpi-tour &&\
        zip -r wpi-tour-linux.zip wpi-tour -x \*CVS\* &&\
        rm -rf wpi-tour

dist-win32: wpi-tour.exe
    mkdir -p dist/wpi-tour
    cd dist &&\
```

```
        ln -s ../../cursors wpi-tour &&\
        ln -s ../../data.xml wpi-tour &&\
        ln -s ../../slides wpi-tour &&\
        ln -s ../../wpi-tour.exe wpi-tour &&\
        cp ../win-libs/*.dll wpi-tour &&\
        zip -r wpi-tour-win32.zip wpi-tour -x \*CVS\* &&\
        rm -rf wpi-tour

dist-data:
    zip -ur wpi-tour-data.zip slides; ret=$$?; \
    if [[ $$ret = 12 || $$ret = 0 ]]; then true else false; fi
    mv wpi-tour-data.zip dist

dist-linux-nodata: wpi-tour
    mkdir -p dist/wpi-tour
    cd dist &&\
        ln -s ../../cursors wpi-tour &&\
        ln -s ../../data.xml wpi-tour &&\
        ln -s ../../wpi-tour wpi-tour &&\
        zip -r wpi-tour-linux-nodata.zip wpi-tour -x \*CVS\* &&\
        rm -rf wpi-tour

dist-win32-nodata: wpi-tour.exe
    mkdir -p dist/wpi-tour
    cd dist &&\
        ln -s ../../cursors wpi-tour &&\
        ln -s ../../data.xml wpi-tour &&\
        ln -s ../../wpi-tour.exe wpi-tour &&\
        cp ../win-libs/*.dll wpi-tour &&\
        zip -r wpi-tour-win32-nodata.zip wpi-tour -x \*CVS\* &&\
        rm -rf wpi-tour

# $Revision: 1.9 $
```

XML manipulation library

Listing A.5: `xmlnode.c`

```
/**
 * @file xmlnode.c XML DOM functions
 *
 * gaim
 *
 * Gaim is the legal property of its developers, whose names are too numerous
 * to list here.  Please refer to the COPYRIGHT file distributed with this
 * source distribution.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

/* A lot of this code at least resembles the code in libxode, but since
 * libxode uses memory pools that we simply have no need for, I decided to
 * write my own stuff.  Also, re-writing this lets me be as lightweight
 * as I want to be.  Thank you libxode for giving me a good starting point */

#include <string.h>
#include <glib.h>

#include "xmlnode.h"

#ifdef _WIN32
# define NEWLINE_S "\r\n"
#else
# define NEWLINE_S "\n"
#endif

static xmlnode*
new_node(const char *name, XMLNodeType type)
{
    xmlnode *node = g_new0(xmlnode, 1);

    if(name)
        node->name = g_strdup(name);
    node->type = type;

    return node;
}

xmlnode*
xmlnode_new(const char *name)
{
    g_return_val_if_fail(name != NULL, NULL);

    return new_node(name, XMLNODE_TYPE_TAG);
}

xmlnode *
xmlnode_new_child(xmlnode *parent, const char *name)
{
```

```
    xmlnode *node;

    g_return_val_if_fail(parent != NULL, NULL);
    g_return_val_if_fail(name != NULL, NULL);

    node = new_node(name, XMLNODE_TYPE_TAG);

    xmlnode_insert_child(parent, node);

    return node;
}

void
xmlnode_insert_child(xmlnode *parent, xmlnode *child)
{
    g_return_if_fail(parent != NULL);
    g_return_if_fail(child != NULL);

    child->parent = parent;

    if(parent->lastchild) {
        parent->lastchild->next = child;
    } else {
        parent->child = child;
    }

    parent->lastchild = child;
}

void
xmlnode_insert_data(xmlnode *node, const char *data, gssize size)
{
    xmlnode *child;
    gsize real_size;

    g_return_if_fail(node != NULL);
    g_return_if_fail(data != NULL);
    g_return_if_fail(size != 0);

    real_size = size == -1 ? strlen(data) : (gsize)size;

    child = new_node(NULL, XMLNODE_TYPE_DATA);

    child->data = g_memdup(data, real_size);
    child->data_sz = real_size;

    xmlnode_insert_child(node, child);
}

void
xmlnode_remove_attrib(xmlnode *node, const char *attr)
{
    xmlnode *attr_node, *sibling = NULL;

    g_return_if_fail(node != NULL);
    g_return_if_fail(attr != NULL);

    for(attr_node = node->child; attr_node; attr_node = attr_node->next)
    {
        if(attr_node->type == XMLNODE_TYPE_ATTRIB &&
                !strcmp(attr_node->name, attr)) {
            if(node->child == attr_node) {
                node->child = attr_node->next;
            } else if (node->lastchild == attr_node) {
                node->lastchild = sibling;
            } else {
                sibling->next = attr_node->next;
            }
```

```
            xmlnode_free(attr_node);
            return;
        }
        sibling = attr_node;
    }
}

void
xmlnode_set_attrib(xmlnode *node, const char *attr, const char *value)
{
    xmlnode *attrib_node;

    g_return_if_fail(node != NULL);
    g_return_if_fail(attr != NULL);
    g_return_if_fail(value != NULL);

    xmlnode_remove_attrib(node, attr);

    attrib_node = new_node(attr, XMLNODE_TYPE_ATTRIB);

    attrib_node->data = g_strdup(value);

    xmlnode_insert_child(node, attrib_node);
}

const char *
xmlnode_get_attrib(xmlnode *node, const char *attr)
{
    xmlnode *x;

    g_return_val_if_fail(node != NULL, NULL);

    for(x = node->child; x; x = x->next) {
        if(x->type == XMLNODE_TYPE_ATTRIB && !strcmp(attr, x->name)) {
            return x->data;
        }
    }

    return NULL;
}

void
xmlnode_free(xmlnode *node)
{
    xmlnode *x, *y;

    g_return_if_fail(node != NULL);

    x = node->child;
    while(x) {
        y = x->next;
        xmlnode_free(x);
        x = y;
    }

    if(node->name)
        g_free(node->name);
    if(node->data)
        g_free(node->data);
    g_free(node);
}

xmlnode*
xmlnode_get_child(const xmlnode *parent, const char *name)
{
    return xmlnode_get_child_with_namespace(parent, name, NULL);
}
```

```
xmlnode *
xmlnode_get_child_with_namespace(const xmlnode *parent, const char *name, const char *ns)
{
    xmlnode *x, *ret = NULL;
    char **names;
    char *parent_name, *child_name;

    g_return_val_if_fail(parent != NULL, NULL);
    g_return_val_if_fail(name != NULL, NULL);

    names = g_strsplit(name, "/", 2);
    parent_name = names[0];
    child_name = names[1];

    for(x = parent->child; x; x = x->next) {
        const char *xmlns = NULL;
        if(ns)
            xmlns = xmlnode_get_attrib(x, "xmlns");

        if(x->type == XMLNODE_TYPE_TAG && name && !strcmp(parent_name, x->name)
                && (!ns || (xmlns && !strcmp(ns, xmlns)))) {
            ret = x;
            break;
        }
    }

    if(child_name && ret)
        ret = xmlnode_get_child(ret, child_name);

    g_strfreev(names);
    return ret;
}

char *
xmlnode_get_data(xmlnode *node)
{
    GString *str = NULL;
    xmlnode *c;

    g_return_val_if_fail(node != NULL, NULL);

    for(c = node->child; c; c = c->next) {
        if(c->type == XMLNODE_TYPE_DATA) {
            if(!str)
                str = g_string_new("");
            str = g_string_append_len(str, c->data, c->data_sz);
        }
    }

    if (str == NULL)
        return NULL;

    return g_string_free(str, FALSE);
}

static char *
xmlnode_to_str_helper(xmlnode *node, int *len, gboolean formatting, int depth)
{
    GString *text = g_string_new("");
    xmlnode *c;
    char *node_name, *esc, *esc2, *tab = NULL;
    gboolean need_end = FALSE, pretty = formatting;

    g_return_val_if_fail(node != NULL, NULL);

    if(pretty && depth) {
        tab = g_strnfill(depth, '\t');
        text = g_string_append(text, tab);
```

```
    }

    node_name = g_markup_escape_text(node->name, -1);
    g_string_append_printf(text, "<%s", node_name);

    for(c = node->child; c; c = c->next)
    {
        if(c->type == XMLNODE_TYPE_ATTRIB) {
            esc = g_markup_escape_text(c->name, -1);
            esc2 = g_markup_escape_text(c->data, -1);
            g_string_append_printf(text, " %s='%s'", esc, esc2);
            g_free(esc);
            g_free(esc2);
        } else if(c->type == XMLNODE_TYPE_TAG || c->type == XMLNODE_TYPE_DATA) {
            if(c->type == XMLNODE_TYPE_DATA)
                pretty = FALSE;
            need_end = TRUE;
        }
    }

    if(need_end) {
        g_string_append_printf(text, ">%s", pretty ? NEWLINE_S : "");

        for(c = node->child; c; c = c->next)
        {
            if(c->type == XMLNODE_TYPE_TAG) {
                int esc_len;
                esc = xmlnode_to_str_helper(c, &esc_len, pretty, depth+1);
                text = g_string_append_len(text, esc, esc_len);
                g_free(esc);
            } else if(c->type == XMLNODE_TYPE_DATA && c->data_sz > 0) {
                esc = g_markup_escape_text(c->data, c->data_sz);
                text = g_string_append(text, esc);
                g_free(esc);
            }
        }

        if(tab && pretty)
            text = g_string_append(text, tab);
        g_string_append_printf(text, "</%s>%s", node_name, formatting ? NEWLINE_S : "");
    } else {
        g_string_append_printf(text, "/>%s", formatting ? NEWLINE_S : "");
    }

    g_free(node_name);

    if(tab)
        g_free(tab);

    if(len)
        *len = text->len;

    return g_string_free(text, FALSE);
}

char *
xmlnode_to_str(xmlnode *node, int *len)
{
    return xmlnode_to_str_helper(node, len, FALSE, 0);
}

char *
xmlnode_to_formatted_str(xmlnode *node, int *len)
{
    char *xml, *xml_with_declaration;

    g_return_val_if_fail(node != NULL, NULL);
```

```
    xml = xmlnode_to_str_helper(node, len, TRUE, 0);
    xml_with_declaration =
        g_strdup_printf("<?xml version='1.0' encoding='UTF-8' ?>" NEWLINE_S NEWLINE_S "%s
            ", xml);
    g_free(xml);

    return xml_with_declaration;
}

struct _xmlnode_parser_data {
    xmlnode *current;
};

static void
xmlnode_parser_element_start(GMarkupParseContext *context,
        const char *element_name, const char **attrib_names,
        const char **attrib_values, gpointer user_data, GError **error)
{
    struct _xmlnode_parser_data *xpd = user_data;
    xmlnode *node;
    int i;

    if(!element_name) {
        return;
    } else {
        if(xpd->current)
            node = xmlnode_new_child(xpd->current, element_name);
        else
            node = xmlnode_new(element_name);

        for(i=0; attrib_names[i]; i++)
            xmlnode_set_attrib(node, attrib_names[i], attrib_values[i]);

        xpd->current = node;
    }
}

static void
xmlnode_parser_element_end(GMarkupParseContext *context,
        const char *element_name, gpointer user_data, GError **error)
{
    struct _xmlnode_parser_data *xpd = user_data;

    if(!element_name || !xpd->current)
        return;

    if(xpd->current->parent) {
        if(!strcmp(xpd->current->name, element_name))
            xpd->current = xpd->current->parent;
    }
}

static void
xmlnode_parser_element_text(GMarkupParseContext *context, const char *text,
        gsize text_len, gpointer user_data, GError **error)
{
    struct _xmlnode_parser_data *xpd = user_data;

    if(!xpd->current)
        return;

    if(!text || !text_len)
        return;

    xmlnode_insert_data(xpd->current, text, text_len);
}

static GMarkupParser xmlnode_parser = {
```

```
    xmlnode_parser_element_start,
    xmlnode_parser_element_end,
    xmlnode_parser_element_text,
    NULL,
    NULL
};


xmlnode *
xmlnode_from_str(const char *str, gssize size)
{
    struct _xmlnode_parser_data *xpd;
    xmlnode *ret;
    GMarkupParseContext *context;
    gsize real_size;

    g_return_val_if_fail(str != NULL, NULL);

    real_size = size < 0 ? strlen(str) : (gsize)size;
    xpd = g_new0(struct _xmlnode_parser_data, 1);
    context = g_markup_parse_context_new(&xmlnode_parser, 0, xpd, NULL);

    if(!g_markup_parse_context_parse(context, str, real_size, NULL)) {
        while(xpd->current && xpd->current->parent)
            xpd->current = xpd->current->parent;
        if(xpd->current)
            xmlnode_free(xpd->current);
        xpd->current = NULL;
    }
    g_markup_parse_context_free(context);

    ret = xpd->current;
    g_free(xpd);
    return ret;
}

xmlnode *
xmlnode_copy(xmlnode *src)
{
    xmlnode *ret;
    xmlnode *child;
    xmlnode *sibling = NULL;

    g_return_val_if_fail(src != NULL, NULL);

    ret = new_node(src->name, src->type);
    if(src->data) {
        if(src->data_sz) {
            ret->data = g_memdup(src->data, src->data_sz);
            ret->data_sz = src->data_sz;
        } else {
            ret->data = g_strdup(src->data);
        }
    }

    for(child = src->child; child; child = child->next) {
        if(sibling) {
            sibling->next = xmlnode_copy(child);
            sibling = sibling->next;
        } else {
            ret->child = xmlnode_copy(child);
            sibling = ret->child;
        }
        sibling->parent = ret;
    }

    ret->lastchild = sibling;
```

```
    return ret;
}

xmlnode *
xmlnode_get_next_twin(xmlnode *node)
{
    xmlnode *sibling;
    const char *ns = xmlnode_get_attrib(node, "xmlns");

    g_return_val_if_fail(node != NULL, NULL);
    g_return_val_if_fail(node->type == XMLNODE_TYPE_TAG, NULL);

    for(sibling = node->next; sibling; sibling = sibling->next) {
        const char *xmlns = NULL;
        if(ns)
            xmlns = xmlnode_get_attrib(sibling, "xmlns");

        if(sibling->type == XMLNODE_TYPE_TAG && !strcmp(node->name, sibling->name) &&
                (!ns || (xmlns && !strcmp(ns, xmlns))))
            return sibling;
    }

    return NULL;
}

/* $Revision: 1.2 $ */
```

XML manipulation header

Listing A.6: `xmlnode.h`

```
/**
 * @file xmlnode.h XML DOM functions
 * @ingroup core
 *
 * gaim
 *
 * Gaim is the legal property of its developers, whose names are too numerous
 * to list here.  Please refer to the COPYRIGHT file distributed with this
 * source distribution.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */
#ifndef _GAIM_XMLNODE_H_
#define _GAIM_XMLNODE_H_

/**
 * The valid types for an xmlnode
 */
typedef enum _XMLNodeType
{
    XMLNODE_TYPE_TAG,       /**< Just a tag */
    XMLNODE_TYPE_ATTRIB,    /**< Has attributes */
    XMLNODE_TYPE_DATA       /**< Has data */
} XMLNodeType;

/**
 * An xmlnode.
 */
typedef struct _xmlnode
{
    char *name;                 /**< The name of the node. */
    XMLNodeType type;           /**< The type of the node. */
    char *data;                 /**< The data for the node. */
    size_t data_sz;             /**< The size of the data. */
    struct _xmlnode *parent;    /**< The parent node or @c NULL.*/
    struct _xmlnode *child;     /**< The child node or @c NULL.*/
    struct _xmlnode *lastchild; /**< The last child node or @c NULL.*/
    struct _xmlnode *next;      /**< The next node or @c NULL. */
} xmlnode;

/**
 * Creates a new xmlnode.
 *
 * @param name The name of the node.
 *
 * @return The new node.
 */
xmlnode *xmlnode_new(const char *name);

/**
 * Creates a new xmlnode child.
```

```
 *
 * @param parent The parent node.
 * @param name   The name of the child node.
 *
 * @return The new child node.
 */
xmlnode *xmlnode_new_child(xmlnode *parent, const char *name);

/**
 * Inserts a node into a node as a child.
 *
 * @param parent The parent node to insert child into.
 * @param child  The child node to insert into parent.
 */
void xmlnode_insert_child(xmlnode *parent, xmlnode *child);

/**
 * Gets a child node named name.
 *
 * @param parent The parent node.
 * @param name   The child's name.
 *
 * @return The child or NULL.
 */
xmlnode *xmlnode_get_child(const xmlnode *parent, const char *name);

/**
 * Gets a child node named name in a namespace.
 *
 * @param parent The parent node.
 * @param name   The child's name.
 * @param xmlns  The namespace.
 *
 * @return The child or NULL.
 */
xmlnode *xmlnode_get_child_with_namespace(const xmlnode *parent, const char *name, const
    char *xmlns);

/**
 * Gets the next node with the same name as node.
 *
 * @param node The node of a twin to find.
 *
 * @return The twin of node or NULL.
 */
xmlnode *xmlnode_get_next_twin(xmlnode *node);

/**
 * Inserts data into a node.
 *
 * @param node   The node to insert data into.
 * @param data   The data to insert.
 * @param size   The size of the data to insert.  If data is
 *               null-terminated you can pass in -1.
 */
void xmlnode_insert_data(xmlnode *node, const char *data, gssize size);

/**
 * Gets data from a node.
 *
 * @param node The node to get data from.
 *
 * @return The data from the node.
 */
char *xmlnode_get_data(xmlnode *node);

/**
 * Sets an attribute for a node.
```

```
130  *
     * @param node  The node to set an attribute for.
     * @param attr  The name of the attribute.
     * @param value The value of the attribute.
     */
135 void xmlnode_set_attrib(xmlnode *node, const char *attr, const char *value);

    /**
     * Gets an attribute from a node.
     *
140  * @param node The node to get an attribute from.
     * @param attr The attribute to get.
     *
     * @return The value of the attribute.
     */
145 const char *xmlnode_get_attrib(xmlnode *node, const char *attr);

    /**
     * Removes an attribute from a node.
     *
150  * @param node The node to remove an attribute from.
     * @param attr The attribute to remove.
     */
    void xmlnode_remove_attrib(xmlnode *node, const char *attr);

155 /**
     * Returns the node in a string of xml.
     *
     * @param node The starting node to output.
     * @param len  Address for the size of the string.
160  *
     * @return The node represented as a string.  You must
     *         g_free this string when finished using it.
     */
    char *xmlnode_to_str(xmlnode *node, int *len);
165
    /**
     * Returns the node in a string of human readable xml.
     *
     * @param node The starting node to output.
170  * @param len  Address for the size of the string.
     *
     * @return The node as human readable string including
     *         tab and new line characters.  You must
     *         g_free this string when finished using it.
175  */
    char *xmlnode_to_formatted_str(xmlnode *node, int *len);

    /**
     * Creates a node from a string of XML.  Calling this on the
180  * root node of an XML document will parse the entire document
     * into a tree of nodes, and return the xmlnode of the root.
     *
     * @param str  The string of xml.
     * @param size The size of the string, or -1 if @a str is
185  *             NUL-terminated.
     *
     * @return The new node.
     */
    xmlnode *xmlnode_from_str(const char *str, gssize size);
190
    /**
     * Creates a new node from the source node.
     *
     * @param src The node to copy.
195  *
     * @return A new copy of the src node.
     */
```

```
xmlnode *xmlnode_copy(xmlnode *src);

/**
 * Frees a node and all of it's children.
 *
 * @param node The node to free.
 */
void xmlnode_free(xmlnode *node);

/* $Revision: 1.2 $ */

#endif /* _GAIM_XMLNODE_H_ */
```

DTD describing a format to define a game

Listing A.7: `game.dtd`

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT game (slide+)>

<!ELEMENT slide     (file, action?, next?)>
<!ELEMENT file      (#PCDATA)>

<!ELEMENT action    (state+)>
<!ELEMENT next      (left|right|up|down|forward|left180|right180)+>

<!ELEMENT left      (#PCDATA)>
<!ELEMENT left180   (#PCDATA)>
<!ELEMENT right     (#PCDATA)>
<!ELEMENT right180  (#PCDATA)>
<!ELEMENT up        (#PCDATA)>
<!ELEMENT down      (#PCDATA)>
<!ELEMENT forward   (#PCDATA)>

<!ELEMENT state     (#PCDATA)>

<!ATTLIST action
    left    CDATA    #REQUIRED
    right   CDATA    #REQUIRED
    top     CDATA    #REQUIRED
    bottom  CDATA    #REQUIRED
    type    (moore)  #REQUIRED>
```

Script to draw a directed graph of all paths

Listing A.8: `path-to-graph.pl`

```
#!/usr/bin/perl -w
use strict;
use Spreadsheet::ParseExcel;
use GraphViz;

my $cluster = 0;

my $g = GraphViz->new(  rankdir=>1,
                        node=>{shape=>'box'},
                        edge=>{fontsize=>8},
                        epsilon=>'.0000000001',
                        no_overlap=>'true',
                        layout=>'dot',
                        splines=>'true',
                        concentrate=>$cluster,
                        graph=>{remincross=>1},
                        nodesep=>1,
);

foreach my $doc (@ARGV) {
    my $exl = new Spreadsheet::ParseExcel::Workbook->Parse($doc);

    foreach my $sheet (@{$exl->{Worksheet}}) {
        for(my $iR = $sheet->{MinRow}; defined $sheet->{MaxRow} && $iR <= $sheet->{MaxRow}
             ; $iR++) {
            my ($pic, $forward, $left, $right, $spin) = map {defined $_ ? $_->Value :
                undef} @{$sheet->{Cells}[$iR]};
            if (not defined $pic or $pic eq 'picture' or $pic eq 'pic') {
                next;
            }

            if ($cluster) {
                $g->add_node($pic, cluster=>{name=>$doc, style=>'filled', fillcolor=>'
                    lightgray'});
            } else {
                $g->add_node($pic);
            }

            warn "$doc: $pic: left, right, and turn defined.\n" if defined $left and
                defined $right and defined $spin;
            warn "$doc: $pic: Endpoint slide.\n" if not defined $left and not defined
                $right and not defined $spin and not defined $forward;

            $g->add_edge($pic, $forward, label=>'Forward') if defined $forward;

            if (defined $left) {
                $g->add_edge($pic, $left, label=>'Left');
            } elsif (defined $spin) {
                $g->add_edge($pic, $spin, label=>'Left180');
            }
            if (defined $right) {
                $g->add_edge($pic, $right, label=>'Right');
            } elsif (defined $spin) {
                $g->add_edge($pic, $spin, label=>'Right180');
            }
        }
    }
}

binmode(STDOUT);
print $g->as_fig;
warn $g->as_canon;
```

Script to convert Excel-format graphs into XML

Listing A.9: `path-to-xml.pl`

```
#!/usr/bin/perl -w
use strict;
use Spreadsheet::ParseExcel;

print <<HERE;
<game>
    <config>
        <screensize width='1024' height='768'/>
        <slidesize width='800' height='600'/>
        <color is='24'/>
        <fullscreen is='0'/>
        <testmode is='1'/>
        <bgcolor is='#000000'/>
        <transition is='best'/>
        <zip allowed='0' start='0'/>
        <paths>
            <text>data/text</text>
            <objects>data/objects</objects>
            <slides>data/slides</slides>
            <sounds>data/sounds</sounds>
            <images>data/images</images>
            <movies>data/movies</movies>
        </paths>
    </config>
HERE

foreach my $doc (@ARGV) {
    my $exl = new Spreadsheet::ParseExcel::Workbook->Parse($doc);

    foreach my $sheet (@{$exl->{Worksheet}}) {
        for(my $iR = $sheet->{MinRow}; defined $sheet->{MaxRow} && $iR <= $sheet->{MaxRow}
             ; $iR++) {
            my ($pic, $forward, $left, $right, $spin) = map {defined $_ ? $_->Value :
                undef} @{$sheet->{Cells}[$iR]};
            if (not defined $pic or $pic eq 'picture' or $pic eq 'pic') {
                #if (defined $pic) {
                #    no warnings;
                #    warn "Header?: (",
                #            join(',', qw/pic forward left right spin/), ") (",
                #            join(',', ($pic,$forward,$left,$right,$spin)), ")\n";
                #    use warnings;
                #}
                next;
            }

            warn "$doc: $pic: left, right, and turn defined.\n" if defined $left and
                defined $right and defined $spin;
            warn "$doc: $pic: Endpoint slide.\n" if not defined $left and not defined
                $right and not defined $spin and not defined $forward;

            print "    <slide>\n",
                  "        <file>$pic</file>\n",
                  "        <next>\n";
            print "            <forward>$forward</forward>\n" if defined $forward;

            if (defined $left) {
                print "            <left>$left</left>\n"
            } elsif (defined $spin) {
                print "            <left180>$spin</left180>\n"
            }
            if (defined $right) {
                print "            <right>$right</right>\n"
            } elsif (defined $spin) {
```

```
                print "            <right180>$spin</right180>\n"
            }
            print "        </next>\n    </slide>\n";
        }
    }
}
print "</game>\n";
```

Script to convert XML to Pyzzle's input format

Listing A.10: `xml2py.pl`

```
#!/usr/bin/perl -w
use strict;

use XML::Simple;
use Data::Dumper qw/Dumper/;

my $xp = XML::Simple->new(ForceArray=>[qw/slide state/],
                          GroupTags=>{
                              color      => 'is',
                              fullscreen => 'is',
                              testmode   => 'is',
                              bgcolor    => 'is',
                              transition => 'is',
                          },
);

our ($file, $prefix) = ('', '');
sub out(@) {
    $file .= $prefix;
    $file .= $_ foreach @_;
}

my $lname = &{sub {my $count = 0; return sub {return "lFunc" . $count++;}}};

my $x = $xp->XMLin(shift);

my $c = $x->{config};

#die Dumper $x;

out <<HERE;
import os, sys
import pyzzle
from pyzzle import *
parameters.setScreenSize(($c->{screensize}->{width},$c->{screensize}->{height}))
parameters.setSlideSize(($c->{slidesize}->{width},$c->{slidesize}->{height}))
parameters.setResolution($c->{color})
parameters.setFullScreen($c->{fullscreen})
parameters.setTestMode($c->{testmode})
parameters.setDefaultTransition($c->{transition}_transition)
parameters.setZipMode($c->{zip}->{start})

X1 = 0
X2 = 2
Y1 = 1
Y2 = 3
WIDTH = 0
HEIGHT = 1
T = parameters.TURNMAPDEPTH

SIZE      = (parameters.SLIDESIZE[0], parameters.SLIDESIZE[1])
LEFTRECT     = (0, 0, T, SIZE[HEIGHT])
LEFT180RECT  = LEFTRECT
RIGHTRECT    = (SIZE[WIDTH]-T, 0, T,SIZE[HEIGHT])
RIGHT180RECT = RIGHTRECT
UPRECT    = (0, 0, SIZE[WIDTH], T)
DOWNRECT  = (0, SIZE[HEIGHT]-T, SIZE[WIDTH], SIZE[HEIGHT])
FORWARDRECT = ( LEFTRECT[X2]+1, UPRECT[Y2]+1,     RIGHTRECT[X1]-1, SIZE[HEIGHT]/2-1)
BACKRECT    = ( LEFTRECT[X2]+1, SIZE[HEIGHT]/2+1, RIGHTRECT[X1]-1, DOWNRECT[Y1]-1)

mustDie = 0
def newSlide(file):
    global mustDie
```

```
    path = os.path.join(paths.getSlidePath(), file + '.jpg');
    if not os.access(path, os.F_OK):
        print "Eek!", file, "is missing"
        mustDie = 1
    return Slide();

HERE

out "parameters.disableZipMode()\n" if $c->{zip}->{allowed} == 0;

while (my ($key, $val) = each %{$c->{paths}}) {
    $key =~ s/(.*?)(s|)$/\u$1/;
    out "paths.set${key}Path(os.path.join(",
        join(',', (map {"'$_'"} split('/', $val))), "))\n";
}

my %slides;
foreach my $s (@{$x->{slide}}) {
    local $prefix = $s->{file};
    out " = newSlide(\'$s->{file}\')\n";
    $slides{$s->{file}}++;
}

$c->{bgcolor} =~ m/#(..)(..)(..)/;
out "parameters.setBackgroundColor((",join(',', map(hex, $1, $2, $3)),"))\n";

sub noType($$;$)
{
    my ($fn, $n, $quiet) = (@_);
    $quiet = 0 unless defined $quiet;
    my %n = %{$n};
    warn "No nav type could be guessed from (", join(',', keys %n), ")\n"
        if not $quiet;
    while (my ($key, $val) = each %n) {
        if (defined $slides{$val}) {
            out ".appendHotspot($val, ", uc($key)."RECT, ", "$key", "_cursor)\n";
        } else {
            warn "Cannot create $key transition from $fn to unknown slide: $val (skipping)
                \n"
        }
    }
}

sub numCheck($$$)
{
    my ($guess, $n, $cnt) = (shift, shift, shift);
    my @k = keys %{$n};
    warn "Direction ignored in $guess guess from (", join(',', @k), ")\n"
        if @k != $cnt;
}

foreach my $s (@{$x->{slide}}) {
    my $self = $s->{file};
    if (defined $s->{action}) {
        my $a = $s->{action};
        die "Can't handle action type $a->{type}\n"
            unless defined $a->{type} and $a->{type} eq 'moore';
        my $fname = &$lname;
        my $if = 'if';
        out "def $fname():\n";
        while (my ($id, $data) = each %{$a->{state}}) {
            out "    $if $self.getState() == $id:\n",
                "        $self.updateSlide('$data->{content}.jpg')\n",
                "        $self.setState($data->{next})\n";
            $if = 'elif' if $if eq 'if';
        }
        out "    else:\n",
            "        $self.updateSlide('$self.jpg')\n",
```

```
            "        $self.setState(1)\n";
        $_ =~ s#(.*)%#($1/100).'*SIZE[0]'#e foreach @$a{qw/left right/};
        $_ =~ s#(.*)%#($1/100).'*SIZE[1]'#e foreach @$a{qw/top bottom/};
        $a->{right}  .= "-$a->{left}";
        $a->{bottom} .= "-$a->{top}";
        out "$self.appendHotspot(action,(",
             join(',', @$a{qw/left top right bottom/}), "),action_cursor)\n";
        out "$self.setState(1)\n",
            "$self.setAction($fname)\n";
    }
    local $prefix = $s->{file};
    out ".setSlideFile('$s->{file}.jpg')\n";

    my %n = %{$s->{next}};
    if (defined $s->{action})
    {
        noType($s->{file}, \%n, 1);
    }
    elsif (defined $n{forward})
    {
        if (defined $n{left180} and defined $n{right180})
        {
            out ".setNavType('std')\n";
            out ".setNavigation([", join(',', @n{qw/left180 forward right180/}),"])\n";
            numCheck('std180', \%n, 3);
        }
        elsif (defined $n{left} and defined $n{right})
        {
            out ".setNavType('std')\n";
            out ".setNavigation([", join(',', @n{qw/left forward right/}),"])\n";
            numCheck('std', \%n, 3);
        }
        else {
            noType($s->{file}, \%n);
        }
    }
    elsif (defined $n{left} and defined $n{right})
    {
        if (defined $n{up})
        {
            out ".setNavType('LandRup')\n";
            out ".setNavigation([", join(',', @n{qw/left up right/}),"])\n";
            numCheck('LandRup', \%n, 3);
        }
        elsif (defined $n{down})
        {
            out ".setNavType('LandRdown')\n";
            out ".setNavigation([", join(',', @n{qw/left down right/}),"])\n";
            numCheck('LandRdown', \%n, 3);
        }
        else
        {
            out ".setNavType('LandR')\n";
            out ".setNavigation([", join(',', @n{qw/left right/}),"])\n";
            numCheck('LandR', \%n, 2);
        }

    }
    elsif (defined $n{up} and defined $n{down})
    {
        out ".setNavType('updown')\n";
        out ".setNavigation([", join(',', @n{qw/up down/}),"])\n";
        numCheck('updown', \%n, 2);
    }
    else
    {
        noType($s->{file}, \%n);
    }
```

```
    $prefix='';
    out "\n";
}

out <<HERE;
pyzzle.setFirstSlide($x->{slide}->[0]->{file})

if mustDie:
    print 'Bailing out due to previous errors!'
    sys.exit(-1)

pyzzle.start()
HERE

print $file, "\n";
```

LaTeX Class For WPI Reports

Listing A.11: `wpi.cls`

```
\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\def\RCS$#1: #2 ${\expandafter\def\csname RCS#1\endcsname{#2}}
\RCS$Revision: 1.4 $
\RCS$Date: 2006/04/28 14:20:33 $

\ProvidesClass{wpi}
              [\RCSDate\space v\RCSRevision\space
               (WPI M/IQP document class)]

\newcommand\@ptsize{}

\setlength\paperheight {11in}
\setlength\paperwidth  {8.5in}
\DeclareOption{landscape}
   {\setlength\@tempdima   {\paperheight}%
    \setlength\paperheight {\paperwidth}%
    \setlength\paperwidth  {\@tempdima}}

\newif\if@mqp
\newif\if@appendix
\newif\if@magictoc

\@mqptrue
\@appendixfalse
\@magictocfalse

\DeclareOption{mqp}{\@mqptrue}
\DeclareOption{iqp}{\@mqpfalse}
\DeclareOption{10pt}{\renewcommand\@ptsize{0}}
\DeclareOption{11pt}{\renewcommand\@ptsize{1}}
\DeclareOption{12pt}{\renewcommand\@ptsize{2}}
\DeclareOption{draft}{\setlength\overfullrule{5pt}}
\DeclareOption{final}{\setlength\overfullrule{0pt}}
\DeclareOption{leqno}{\input{leqno.clo}}
\DeclareOption{fleqn}{\input{fleqn.clo}}
\ExecuteOptions{11pt,final,mqp}
\ProcessOptions
\input{size1\@ptsize.clo}
\setlength\lineskip{1\p@}
\setlength\normallineskip{1\p@}
\renewcommand\baselinestretch{}
\setlength\parskip{0\p@ \@plus \p@}
\@lowpenalty   51
\@medpenalty  151
\@highpenalty 301
\setcounter{topnumber}{2}
\renewcommand\topfraction{.7}
\setcounter{bottomnumber}{1}
\renewcommand\bottomfraction{.3}
\setcounter{totalnumber}{3}
\renewcommand\textfraction{.2}
\renewcommand\floatpagefraction{.5}
\setcounter{dbltopnumber}{2}
\renewcommand\dbltopfraction{.7}
\renewcommand\dblfloatpagefraction{.5}

\RequirePackage{setspace}
\RequirePackage[letterpaper,margin=1in,lmargin=1.5in]{geometry}
\RequirePackage{ifthen}

\def\title#1{\gdef\@title{{\large\textsc{\textbf{#1}}}}\gdef\mytitle{{#1}}}
\def\date#1{\gdef\@date{Date:~#1}\gdef\mydate{{\@date}}}
\def\id#1{\gdef\@id{#1}\gdef\myid{{#1}}}
```

```
% This creates a list type.
%   #1: name
%   #2: what \and expands to
%   #3: prefix
\long\def\listtype#1#2#3{
    \expandafter\gdef\csname#1\endcsname##1{{%
        \expandafter\gdef\csname @#1\endcsname{{%
            \def\and{#2}
            #3##1%
        }}%
        \def\and{and }
        \expandafter\xdef\csname my#1\endcsname{{##1}}
    }}
}

\def\@sign{\rule{2in}{0.8pt}\\[\@rskip]}

\listtype{author}{\par\@sign}{\@sign}
\listtype{keyword}{\item}{\item}
\listtype{advisor}{\par\@sign}{\@sign}

\def\@tskip{0.3in}
\def\@rskip{.05in}

\newcommand{\maketitle}{{%
    \interlinepenalty \@M
    \linepenalty \@M
    \hyphenpenalty \@M
    \exhyphenpenalty \@M
    \offinterlineskip
    \begin{titlepage}
    \singlespacing
    \setlength{\parskip}{\@tskip minus \@tskip}
    \raggedleft \@id

    \begin{center}

    \@title

    A \if@mqp Major \else Interactive \fi Qualifying Project Report:

    submitted to the Faculty

    of the

    \textsc{Worcester Polytechnic Institute}

    in partial fulfillment of the requirements for the

    Degree of Bachelor of Science

    by

    \@author

    \@date

    \vspace{\@tskip plus 1fil}

    Approved:

    \end{center}

    \vspace{\@tskip}

    \hbox to \hsize{%
```

```
        \rlap{\vbox{%
            \begin{enumerate*}
                \parskip 0pt
                \itemsep 3pt
                \@keyword
            \end{enumerate*}
        }}%
        \hss
        \llap{\vbox{%
            \@advisor
        }}%
    }

    \end{titlepage}
}}

\newcommand*\chaptermark[1]{}

\setcounter{secnumdepth}{2}
\newcounter {chapter}
\newcounter {section}[chapter]
\newcounter {subsection}[section]
\newcounter {subsubsection}[subsection]
\renewcommand \thechapter {\@arabic\c@chapter}
\renewcommand \thesection {\thechapter.\@arabic\c@section}
\renewcommand\thesubsection   {\thesection.\@arabic\c@subsection}
\renewcommand\thesubsubsection{\thesubsection .\@arabic\c@subsubsection}
\newcommand\@chapapp{}

\newcommand\chapter{\clearpage
                    \thispagestyle{plain}%
                    \global\@topnum\z@
                    \@afterindentfalse
                    \secdef\@chapter\@schapter}
\def\@chapter[#1]#2{\ifnum \c@secnumdepth >\m@ne
                         \refstepcounter{chapter}%
                         \typeout{\@chapapp\space\thechapter.}%
                         \addcontentsline{toc}{chapter}%
                                   {\textsc{\if@appendix \appendixname\space\fi
                                   \protect\numberline{\thechapter%
                                      \if@appendix : \fi}%
                                   }\textsc{#1}}%
                    \else
                      \addcontentsline{toc}{chapter}{\textsc{#1}}%
                    \fi
                    \chaptermark{#1}%
                    \addtocontents{lof}{\protect\addvspace{10\p@}}%
                    \addtocontents{lot}{\protect\addvspace{10\p@}}%
                    \@makechapterhead{#2}%
                    \@afterheading
                    \label{cha:#2}}

\def\@makechapterhead#1{%
  \vspace*{10\p@}%
  {\parindent \z@ \raggedright \normalfont
    \huge\bfseries\textsc{%
    \ifnum \c@secnumdepth >\m@ne
        \if@appendix
            \@chapapp\space \thechapter:
        \else
            \thechapter
        \fi
    \fi
    \space\space #1}\par\nobreak
    \vskip 10\p@
}}

\def\@schapter#1{%
```

```
    \if@magictoc
        \addcontentsline{toc}{chapter}{\textsc{#1}}
    \fi
    \@makeschapterhead{#1}
    \@afterheading
}

\def\@makeschapterhead#1{%
  \vspace*{10\p@}%
  {\parindent \z@ \raggedright
    \normalfont
    \huge \bfseries \textsc{#1}\par\nobreak
    \vskip 10\p@
}}

\newcommand\section{\@startsection {section}{1}{\z@}%
                                   {-3.5ex \@plus -1ex \@minus -.2ex}%
                                   {2.3ex \@plus.2ex}%
                                   {\normalfont\Large\bfseries}}
\newcommand\subsection{\@startsection{subsection}{2}{\z@}%
                                     {-3.25ex\@plus -1ex \@minus -.2ex}%
                                     {1.5ex \@plus .2ex}%
                                     {\normalfont\large\bfseries}}
\newcommand\subsubsection{\@startsection{subsubsection}{3}{\z@}%
                                     {-3.25ex\@plus -1ex \@minus -.2ex}%
                                     {1.5ex \@plus .2ex}%
                                     {\normalfont\normalsize\bfseries}}

\setlength\leftmargini  {2.5em}
\leftmargin  \leftmargini
\setlength\leftmarginii  {2.2em}
\setlength\leftmariniii {1.87em}
\setlength\leftmarginiv  {1.7em}
\setlength\leftmarginv  {1em}
\setlength\leftmarginvi {1em}
\setlength  \labelsep  {.5em}
\setlength  \labelwidth{\leftmargini}
\addtolength\labelwidth{-\labelsep}
\@beginparpenalty -\@lowpenalty
\@endparpenalty   -\@lowpenalty
\@itempenalty     -\@lowpenalty
\renewcommand\theenumi{\@arabic\c@enumi}
\renewcommand\theenumii{\@alph\c@enumii}
\renewcommand\theenumiii{\@roman\c@enumiii}
\renewcommand\theenumiv{\@Alph\c@enumiv}
\newcommand\labelenumi{\theenumi.}
\newcommand\labelenumii{(\theenumii)}
\newcommand\labelenumiii{\theenumiii.}
\newcommand\labelenumiv{\theenumiv.}
\renewcommand\p@enumii{\theenumi}
\renewcommand\p@enumiii{\theenumi(\theenumii)}
\renewcommand\p@enumiv{\p@enumiii\theenumiii}
\newcommand\labelitemi{\textbullet}
\newcommand\labelitemii{\normalfont\bfseries \textendash}
\newcommand\labelitemiii{\textasteriskcentered}
\newcommand\labelitemiv{\textperiodcentered}
\newenvironment{description}
               {\list{}{\labelwidth\z@ \itemindent-\leftmargin
                        \let\makelabel\descriptionlabel}}
               {\endlist}
\newcommand*\descriptionlabel[1]{\hspace\labelsep
                                 \normalfont\bfseries #1}

\newenvironment{abslike}[1]{%
    \titlepage
    \null\vfil
    \@beginparpenalty\@lowpenalty
    \begin{center}%
```

```
      \bfseries #1
      \@endparpenalty\@M
    \end{center}
    }%
    {\par\vfil\null\endtitlepage%
}

\newenvironment{abstract}{\abslike{Abstract}}{\endabslike}
\newenvironment{ack}{\abslike{Acknowledgements}}{\endabslike}

\newenvironment{verse}
               {\let\\\@centercr
                \list{}{\itemsep      \z@
                        \itemindent   -1.5em%
                        \listparindent\itemindent
                        \rightmargin  \leftmargin
                        \advance\leftmargin 1.5em}%
                \item\relax}
               {\endlist}
\newenvironment{quotation}
               {\list{}{\listparindent 1.5em%
                        \itemindent    \listparindent
                        \rightmargin   \leftmargin
                        \parsep        \z@ \@plus\p@}%
                \item\relax}
               {\endlist}
\newenvironment{quote}
               {\list{}{\rightmargin\leftmargin}%
                \item\relax}
               {\endlist}
\newenvironment{titlepage}
    {\newpage
     \thispagestyle{empty}%
     \setcounter{page}\@ne}%
    {\newpage\setcounter{page}\@ne}

\newcommand\appendix{\par
  \setcounter{chapter}{0}%
  \setcounter{section}{0}%
  \gdef\@chapapp{\appendixname}%
  \gdef\thechapter{\@Alph\c@chapter}
  \@appendixtrue
}
\setlength\arraycolsep{5\p@}
\setlength\tabcolsep{6\p@}
\setlength\arrayrulewidth{.4\p@}
\setlength\doublerulesep{2\p@}
\setlength\tabbingsep{\labelsep}
\skip\@mpfootins = \skip\footins
\setlength\fboxsep{3\p@}
\setlength\fboxrule{.4\p@}
\@addtoreset {equation}{chapter}
\renewcommand\theequation
  {\ifnum \c@chapter>\z@ \thechapter.\fi \@arabic\c@equation}
\newcounter{figure}[chapter]
\renewcommand \thefigure
     {\ifnum \c@chapter>\z@ \thechapter.\fi \@arabic\c@figure}
\def\fps@figure{tbp}
\def\ftype@figure{1}
\def\ext@figure{lof}
\def\fnum@figure{\figurename\nobreakspace\thefigure}
\newenvironment{figure}
               {\@float{figure}}
               {\end@float}
\newenvironment{figure*}
               {\@dblfloat{figure}}
               {\end@dblfloat}
\newcounter{table}[chapter]
```

```
\renewcommand \thetable
     {\ifnum \c@chapter>\z@ \thechapter.\fi \@arabic\c@table}
\def\fps@table{tbp}
\def\ftype@table{2}
\def\ext@table{lot}
\def\fnum@table{\tablename\nobreakspace\thetable}
\newenvironment{table}
               {\@float{table}}
               {\end@float}
\newenvironment{table*}
               {\@dblfloat{table}}
               {\end@dblfloat}
\newlength\abovecaptionskip
\newlength\belowcaptionskip
\setlength\abovecaptionskip{10\p@}
\setlength\belowcaptionskip{0\p@}
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{#1: #2}%
  \ifdim \wd\@tempboxa >\hsize
    #1: #2\par
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}
\DeclareOldFontCommand{\bf}{\normalfont\bfseries}{\mathbf}
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
\DeclareOldFontCommand{\sl}{\normalfont\slshape}{\@nomath\sl}
\DeclareOldFontCommand{\sc}{\normalfont\scshape}{\@nomath\sc}
\DeclareRobustCommand*\cal{\@fontswitch\relax\mathcal}
\DeclareRobustCommand*\mit{\@fontswitch\relax\mathnormal}
\newcommand\@pnumwidth{1.55em}
\newcommand\@tocrmarg{2.55em}
\newcommand\@dotsep{4.5}
\setcounter{tocdepth}{2}
\newcommand\tableofcontents{%
    \chapter*{\contentsname
        \@mkboth{%
           \MakeUppercase\contentsname}{}}%
    \@starttoc{toc}%
    }
\newcommand*\l@chapter[2]{%
  \ifnum \c@tocdepth >\m@ne
    \addpenalty{-\@highpenalty}%
    \vskip 1.0em \@plus\p@
    \setlength\@tempdima{1.5em}%
    \begingroup
      \parindent \z@ \rightskip \@pnumwidth
      \parfillskip -\@pnumwidth
      \leavevmode \bfseries
      \advance\leftskip\@tempdima
      \hskip -\leftskip
      #1\nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss #2}\par
      \penalty\@highpenalty
    \endgroup
  \fi}
\newcommand*\l@section{\@dottedtocline{1}{1.5em}{2.3em}}
\newcommand*\l@subsection{\@dottedtocline{2}{3.8em}{3.2em}}
\newcommand*\l@subsubsection{\@dottedtocline{3}{7.0em}{4.1em}}
\newcommand\listoffigures{%
    \@magictoctrue
    \chapter*{\listfigurename}
    \@magictocfalse
      \@mkboth{\MakeUppercase\listfigurename}%
```

```
                {}%
    \@starttoc{lof}%
    }
\newcommand*\l@figure{\@dottedtocline{1}{1.5em}{2.3em}}
\newcommand\listoftables{%
    \@magictoctrue
    \chapter*{\listtablename}
    \@magictocfalse
      \@mkboth{%
          \MakeUppercase\listtablename}%
         {}%
    \@starttoc{lot}%
    }
\let\l@table\l@figure
\newdimen\bibindent
\setlength\bibindent{1.5em}
\newenvironment{thebibliography}[1]
     {\@magictoctrue
      \chapter*{\bibname}%
      \@magictocfalse
      \@mkboth{\MakeUppercase\bibname}{}%
      \list{\@biblabel{\@arabic\c@enumiv}}%
           {\settowidth\labelwidth{\@biblabel{#1}}%
            \leftmargin\labelwidth
            \advance\leftmargin\labelsep
            \@openbib@code
            \usecounter{enumiv}%
            \let\p@enumiv\@empty
            \renewcommand\theenumiv{\@arabic\c@enumiv}}%
      \sloppy
      \clubpenalty4000
      \@clubpenalty \clubpenalty
      \widowpenalty4000%
      \sfcode`\.\@m}
     {\def\@noitemerr
       {\@latex@warning{Empty `thebibliography' environment}}%
      \endlist}
\newcommand\newblock{\hskip .11em\@plus.33em\@minus.07em}
\let\@openbib@code\@empty

\AtBeginDocument{%
    \def\MakeUppercase{}
    \let\@OldBibEnd\endthebibliography
    \renewcommand{\endthebibliography}{%
        \@OldBibEnd
        \addcontentsline{toc}{chapter}{\textsc{\bibname}}}%
    \renewcommand\chaptermark[1]{%
        \markboth{%
            \if@appendix%
                \@chapapp\ \thechapter:\space\space#1
            \else
                #1
            \fi
        }{}
    }%
    \renewcommand\sectionmark[1]{%
        \markright{#1}
    }%
    \let\@OldMkBoth\@mkboth
    \def\@mkboth#1#2{%
        \ifthenelse{\equal{#1}{#2}}%
            {\@OldMkBoth{#1}{}}%
            {\@OldMkBoth{#1}{#2}}%
    }%
}

\newenvironment{theindex}
               {
```

```
                \@makeschapterhead{\indexname}
                \@mkboth{\MakeUppercase\indexname}%
                        {}%
                \thispagestyle{plain}\parindent\z@
                \parskip\z@ \@plus .3\p@\relax
                \columnseprule \z@
                \columnsep 35\p@
                \let\item\@idxitem}
               {\clearpage}
\newcommand\@idxitem{\par\hangindent 40\p@}
\newcommand\subitem{\@idxitem \hspace*{20\p@}}
\newcommand\subsubitem{\@idxitem \hspace*{30\p@}}
\newcommand\indexspace{\par \vskip 10\p@ \@plus5\p@ \@minus3\p@\relax}
\renewcommand\footnoterule{%
  \kern-3\p@
  \hrule\@width.4\columnwidth
  \kern2.6\p@}
\@addtoreset{footnote}{chapter}
\newcommand\@makefntext[1]{%
    \parindent 1em%
    \noindent
    \hb@xt@1.8em{\hss\@makefnmark}#1}
\newcommand\contentsname{Contents}
\newcommand\listfigurename{List of Figures}
\newcommand\listtablename{List of Tables}
\newcommand\bibname{Bibliography}
\newcommand\indexname{Index}
\newcommand\figurename{Figure}
\newcommand\tablename{Table}
\newcommand\chaptername{Chapter}
\newcommand\appendixname{Appendix}
\newcommand\abstractname{Abstract}
\def\today{\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space\number\day, \number\year}
\setlength\columnsep{10\p@}
\setlength\columnseprule{0\p@}
\pagestyle{plain}
\pagenumbering{arabic}

\RequirePackage{listings}
\RequirePackage{url}

% Code listing setup
\lstset{numbers=left,numberstyle=\tiny,tabsize=4,
        basicstyle=\scriptsize\ttfamily,frame=L,breaklines=true,stepnumber=5,firstnumber
            =0}

\def\@codedir{}
\newcommand{\codedir}[1]{%
    \gdef\@codedir{#1}
}
\def\thecodedir{\@codedir}

% \code[path]{filename}{Title}
\newcommand{\code}[3][\@codedir]{{%
    \ifthenelse{\equal{#1}{}}{\def\@path{#2}}{\def\@path{#1/#2}}
    \singlespacing
    \setcounter{secnumdepth}{-1}%
    \section{#3}%
    \urldef\@EvilHack\path{#2}%
    \lstinputlisting[label=code:#2,caption=\@EvilHack]{\@path}%
    \newpage
    \setcounter{secnumdepth}{2}%
    \doublespacing
}}
```

```
\RequirePackage{calc}

\def\@maxwidth{1}
\def\maxwidth#1{\def\@maxwidth{#1}}

\def\picwidth{%
    \ifdim\Gin@nat@width>\@maxwidth\linewidth
        \@maxwidth\linewidth
    \else
        \Gin@nat@width
    \fi
}

\def\@figdir{}
\newcommand{\figdir}[1]{%
    \gdef\@figdir{#1}%
}
\def\thefigdir{\@figdir}

\newcommand{\fig}[3][\@figdir]{{%
    \ifthenelse{\equal{#1}{}}{\def\@path{#2}}{\def\@path{#1/#2}}
    \begin{figure}[htbp]%
        \centering
        \includegraphics[width=\picwidth]{\@path}%
        \caption{#3}
        \label{fig:#2}
    \end{figure}%
}}

\newcommand{\figh}[3][\@figdir]{{%
    \ifthenelse{\equal{#1}{}}{\def\@path{#2}}{\def\@path{#1/#2}}
    \begin{figure}[H]%
        \centering
        \includegraphics[width=\picwidth]{\@path}%
        \caption{#3}
        \label{fig:#2}
    \end{figure}%
}}

\RequirePackage{subfigure}

\newcommand{\subfig}[3][0.5in]{%
    \let\OldFig\fig
    \begin{figure}%
        \def\gap{\hspace{#1}}%
        \centering
        \def\fig##1##2{%
            \subfigure[##2]{%
                \label{fig:##1}%
                \includegraphics[width=\picwidth]{\thefigdir/##1}%
            }%
            \gap
        }%
        \def\tab##1##2##3{%
            \subtable[##2]{%
                ##3%
                \label{tab:##1}%
            }%
            \gap
        }%
        \def\endsubfig{%
            \caption{#2}%
            \ifthenelse{\equal{#3}{}}{}{\label{fig:#3}}%
            \let\fig\OldFig
            \end{figure}%
        }%
}
```

```
\raggedbottom
\onecolumn
\doublespacing
\endinput

% vim: set ft=tex:
```

APPENDIX B: DEMOGRAPHIC QUESTIONS

Our project is to create an entertaining and educational virtual tour of the Worcester campus of Worcester Polytechnic Institute. The following survey will ask a variety of questions based on activities you can perform in the game.

E-mail: _____

What age range best describes you?

1. Under 16
2. 14 - 18
3. 19 - 22
4. Over 22

Are you a current or previous student of Worcester Polytechnic Institute?

1. Yes
2. No

Have you ever visited WPI?

1. Yes
2. No

If yes, did you take part in a tour of the campus?

1. Yes
2. No

APPENDIX C: SURVEY

From the start, how would you get from Reunion Plaza to the campus center?

1. Turn right and follow the path to the end and then turn right
2. Go straight until the path ends, turn left and go straight until the end of the path
3. Turn left and enter the Campus center
4. Don't know

Inside of Fuller Labs you can find the _____ Dept.

1. Electrical and Computer Engineering
2. Biology
3. Computer Science
4. Don't know

Salisbury Labs is directly located between Boynton Hall and Atwater Kent?

1. True
2. False
3. Don't know

How would you get from Morgan Hall to Harrington Auditorium?

1. Turn left and follow the path until it goes into Harrington
2. Follow the path straight, after the turn in the path turn left
3. Turn right, follow the path and then turn left
4. Don't know

The Birch Tree is between _____ and _____.

1. Atwater Kent, Goddard Labs
2. Higgins Labs, Alden Memorial
3. Morgan Hall, Daniels Hall

4. Don't know

Alumni Gym is next to Stratton Hall?

1. True
2. False
3. Don't know

How would you get from Olin Hall to Salisbury Labs?

1. Go straight until you enter Salisbury
2. Turn left, follow path to the end turn right and follow path until the end
3. Turn around and enter building
4. Don't know

The Music Dept. is in _____.

1. Bartlet Center
2. Riley Hall
3. Alden Memorial
4. Don't know

Boynton Hall currently holds the administration?

1. True
2. False
3. Don't know

Daniels Hall is connected to _____.

1. Morgan Hall
2. Riley Hall
3. Olin Hall
4. Don't know