CS4911

Team 7

Knights Who Say Ni!

Software Requirements Specification Document

Version: (2.0) **Date: (11/7/2005)**

Table of Contents

1. Introduction

1.1 Purpose

This SRS document is intended as a guide to the functional and non-functional requirements of an improvement to the JES media libraries. This improvement will utilize underlying C library code to improve the performance of the system. John Burton, Andrew Nagel, Sam Gawthrop and Dustin Roberts should use this document to build a new media library for Python. Mark Guzdial should use this document for future development of the media library.

1.2 Scope

We will be recreating the JES media libraries. Currently this system utilizes JAVA libraries to do the actual processing of media files that are resident on the system. The media libraries must be capable of breaking image files of various formats into pixels and display the image file in a pop-up window. The system should also be capable of breaking a sound file into separate individuals samples and playing those files. Both the audio samples and image pixels should be accessible to the users.

The implementation in place now does perform these tasks but it does so using Java libraries, which are rather slow. This project will implement the same functionality using C libraries instead of Java with the hopeful end result being an increase in speed.

1.3 Definitions, Acronyms, and Abbreviations.

API – Application Programmers Interface
CPython – Python implemented in C
GUI – Graphical users interface
Java – Programming language created by Sun Microsystems
JES – Jython Environment for Students
JPEG – algorithm for compressing image files
Jython – Python implemented in Java
PyDoc – Documentation standard for Python
PyGames – Game library for CPython
SDL – Simple DirectMedia Layer
SRS – Software Requirements Specification
Wav – format for audio files
wxPython – GUI framework that is cross platform and available for Python

1.4 References

Project Plan –
http://swiki.cc.gatech.edu:8080/cs4911b-fl05/uploads/132/projectPlan%20v2.pdf

PyDoc –
http://docs.python.org/tut/node6.html#SECTION006760000000000000000

1.5 Overview

The remainder of this document contains an overview of the requirements that have to be in place for the systems on which our implementation will run and the requirements for the implementation itself. Finally the approval process for modification to this document is detailed.

The Overall Description details the way in which the product will fit into the system as a whole and all of the system interfaces will be described. This information is most useful to people who plan on putting this system into actual implementation as the pre-requisites of the system will be entailed.

The Specific Requirements section will detail the specific requirements of the product we are developing. These requirements are those that will be used by the developers to implement the system. The requirements will be useful in developing the Design that will eventually be used to produce the final product.

2. The Overall Description

2.1 Product Perspective

The system that we are building is meant to replace an already existing system called JES. The current system is written using the Java API, which means the Java Virtual Machine is used. Jython itself is also like a virtual machine, interpreting python code to run on Java. This makes anything that is written using JES have to go through 2 interpretations, which slows the processing down. The idea is to replace the media libraries (media.py) in JES with a new Media.py that implements PyGames, which is written in C. If the new implementation uses PyGames, then the python code only has to go through the CPython interpreter, hopefully making it much faster, allowing for more computations to be done in a shorter amount of time.

2.1.1 System Interfaces

There are no external system interfaces that Media.py will interact with.

2.1.2 Interfaces

There is no graphical users interface that will be created for our system. The system is a file that will be imported by CS1315 students to do manipulations of sound and images. The only interface the student will be interacting with is the documentation that we created, that tells them how to use the software and how to install it. There is, however, a requirement for the student to be able to display an image within a GUI. For that, we will use wxPython fulfilling the following requirements:

1. The GUI will be abstracted out of the software required for CS1315, so that any editor may be used.
2. The libraries will be updated to spawn a new window (for text display)
3. The libraries will call CPython functions through the PYGames libraries.
4. The libraries will be platform compatible from MAC OS and Windows.
5. Linux and Unix is not a specific requirement, but since Python is a cross platform language, we will test on Unix and Linux systems in order to let users know of any known issues.

2.1.3 Hardware Interfaces

There are no specific hardware interfaces that will be used.

2.1.4 Software Interfaces

2.1.4.1 CPython version 2.3.5. The system will use version 2.3 of the CPython interpreter available from www.python.org. This interface will be used for interpretation of all python software. Both the user and the developer will use this software for writing any and all programs. The only way to interface with this software is to install it on the computer and use "python <filename>" command to make any python program work.

2.1.4.2 PyGame 1.7.1. The system will implement PyGame version 1.7.1 available from www.pygame.org/download.html. This software will be used for editing both pictures and sound. Although the user will need to install PyGame, they will not have to import it, because Media.py will make all necessary calls.

2.1.4.3 Numeric 23.7. The system will implement Numeric version 23.7 availible from http://coweb.cc.gatech.edu/mediaComp-plan/117. This software will be used for doing complex math functions such as square root. Although the user will need Numeric installed, they will not have to import it in their programs, because Media.py will do that for them.

2.1.4.4 wxPython 2.6.1.0. The system will implement wxPython version 2.6.1.0 available from http://www.wxpython.org/. This software will be used for its file chooser and displaying abilities. Although it is true that PyGame will display files

without wxPython, the customer has shown interest in wxPython because of the ability to do low level activities with custom GUIs. wxPython will not have to be imported into files, as long as they are using the Media.py file.

2.1.5 Communications Interfaces

There are no communication interfaces necessary for this software.

2.1.6 Memory Constraints

Although there are no specific constraints on memory, the user must have enough memory to run their desired operating system, such as Mac OS X or Windows XP. Users should look to their operating system manufacturer to make sure they have enough memory to run the system.

Although there is no specific constraint on memory beside manufacturer requirements, we recommend that all users have at least 256 Megabytes of memory installed, and should use 512 megabytes for optimal performance with most image and sound files.

2.1.7 Operations

The system will run be running anytime a user imports Media.py in their Python programs. This means that it will be at the user's discretion when the system is in use, or when to bring it down. The system may be used unattended if the user writes a script to utilize the system while they are away, but unattended operation is not necessary.

2.1.8 Site Adaptation Requirements

Users will follow the directions and our automated installer in order to install the software in the correct locations for use. It is not required that users install the software at any specific location, but it will be highly recommended that users install at the default locations. This is because any 3rd party Python support will refer to those locations, and any specific locations found within the book by Mark Guzdial will refer to the default location. There are no hardware requirements, although if a user does not have a sound card they may wish to install one to fully realize the potential of the sound manipulation features.

2.2 Product Functions

- The product will replace the current Media.py that was written for JES.
- The product will facilitate the manipulation of images and sound
- The product will provide a simple GUI for the images to be displayed.

2.3 User Characteristics

The user for this system is an average CS1315 student. They are not expected to be computer experts and are not expected to have any programming experience. The user is expected to understand how to do basic functions on a computer such as install software, download software, and know how to move files around a file system.

2.4 Constraints

The only constraint on the system is that it utilizes CPython. We are permitted to use C to write new libraries, but this should not be necessary because PyGame includes the needed functionality.

2.5 Assumptions and Dependencies

This document assumes that wxPython will suffice for creating simple GUIs for python on all platforms (i.e. Mac and Windows).

2.6 Apportioning of Requirements.

All requirements must be completed for this project.

3. Specific Requirements

3.1 External Interfaces

3.1.1 User Interfaces

The only user interface that will be provided for the user when using Media.py is a window provided by wxPython. The user will see this window when they call "show()" in order to display a JPEG image. WxPython will also provide the "pickAFile()" and "pickAFolder()" functionality to the user. This functionality is already built into wxPython, so will only require developers to mask the wxPython function call with a function call in Media.py.

3.1.2 Hardware Interfaces

There are no specific hardware interfaces that are required for the software. It is recommended that users install a sound card that is capable of playing 22000 Hz, 16 bit, mono sounds, in order to utilize the "play()" feature in Media.py, for playing sound files.

3.1.3 Software Interfaces

- CPython version 2.3.5 – CPython will only be loaded when the user calls "python <filename>" from either the windows command line or the Mac OS X terminal. CPython is required to run any python program, and therefore is required for Media.py. CPython will be shut down either when the program running terminates, or the user types "ctrl+c".
- PyGame 1.7.1 – PyGame will provide the decoding, playing, and viewing functions for Media.py. PyGame will be imported into Media.py, so the users will not have to be aware that it exists on the system. Anytime Media.py is used in a Python program, PyGame will be implicitly loaded from the import of PyGame in Media.py. When the program is done with execution or the user presses "ctrl+c", the program will terminate, therefore causing PyGame to terminate.
- Numeric 23.7 – Numeric will provide advanced math functions in Media.py. Numeric will be imported into Media.py, therefore the user will not have to deal directly with it. Anytime Media.py is imported into a Python program, Numeric functionality will also be imported. Numeric will be loaded when any python program imports Media.py and will terminate when that program is finished, or the user presses "ctrl+c".
- wxPython 2.6.1.0 – wxPython is a window handler for Python and will be used for choosing files, folders, and displaying pictures. wxPython will be loaded anytime Media.py is imported into a program, because Media.py will import wxPython. wxPython will terminate when the python program terminates or when the user presses "ctrl+c".

3.2.4 Communications Interfaces

There are no communication interfaces for Media.py.

3.2 Functions

3.2.1 Graphic Modification Feature

The media library must be capable of loading an image from a file by decoding the data and converting it into some kind of an accessible structure. The access structure that will be used to contain the image information is a collection of pixel objects each with its own properties. These pixels will each have levels that can be accessed and modified by the user. In color images there will be three such sets of pixel maps, one for each color.

3.2.1.1 Introduction/Purpose of Feature

The purpose of this feature is to allow the high level interface provided to the end users by the Media.py file to access graphics files and modify those files using a simple API. The system in place for this access currently utilizes Java libraries to control this access. The purpose of this feature is to implement this system using C libraries rather than Java libraries.

3.2.1.2 Stimulus/Response Sequence

The end user will utilize this feature in the following way. First an image file will be created on the fly or loaded from a pre-existing file located on disk. The required supported format is JPEG, but because PyGame supports GIF and BMP, this functionality will be available. The system will load this file into a data structure that contains pixel values for each of the images' color channels. This data structure gives the user direct access to the values contained in the image. This gives the user the capability to modify the image in memory by altering pixel channel values in some way. Finally the user can choose to save the image data back to a disk based file or display it in a GUI window, which is developed as a separate system feature. All files will be saved as a JPEG.

3.2.1.3 Associated Functional Requirements

3.2.1.3.1 Image Load

The system shall be capable of loading images from several different disk based image formats into a graphic object developed as part of the media system. The load will be provided to the python media system by underlying media libraries

3.2.1.3.2 Graphic Object

The system shall include a graphic object that is created by the system as a null image of a specific size or through the loading of an image from a disk file. This graphic object will give the user access to the data contained in the image and data about the image itself. The graphic object will provide the user with access to image size and color information. It shall also provide the user with access to pixel objects that contain the display data for the image itself. The user can modify the data in each color channel.

3.2.1.3.3 Image Save

The system shall be capable of saving the image data contained in the graphic object to a JPEG file. The system will utilize an underlying library to accomplish this functionality.

3.2.2 Audio Editing Feature

3.2.2.1 Introduction/Purpose of feature

This feature will allow users to play and modify several standard audio formats using the Media.py library, which will utilize the PyGame CPython library for the required functionality.

The system shall facilitate opening an existing audio file, and breaking it down into a usable structure, so that the user can mix different samples, speed up, slow down, or do any other type of manipulation to the audio.

3.2.2.2 Stimulus/Response sequence

This feature will be able to play audio for the user to listen to. They will also be able to modify the audio sample using a simple programming interface, and listen to the results of their modifications. The user shall be able to manipulate any audio sample that is supported by PyGame.

3.2.2.3 Associated functional requirements

3.2.2.3.1 Read Audio Files

This feature shall be able to load audio files from the file system. Only WAV files will be supported in Media.py. The system will be able to load any size file, limited only to the size the operating system will allow. However, there is a practical limit on file size, and we will not optimize the system to load files more than 256 megabytes in size.

3.2.2.3.2 Play Audio

This feature shall be able to play audio files after they are loaded and after they have been modified.

3.2.2.3.2 Edit Audio

This feature shall be able to parse an audio file into simple, modifiable samples for the user. Simple means that all wav files will be converted to 22000 Hz, 16 bit, mono sounds. Users will be able to delete, copy, reverse, increase intensity and decrease intensity with any WAV file. These samples shall be accessed as a Python object through any python program that imports media.py

3.2.2.3.3 Save Audio

This feature shall allow the user to save any edited audio files to a disk in WAV format.

3.3 Performance Requirements

There are no specific performance requirements for the system. The idea of the project is to see if there is an increase in speed by utilizing CPython over Jython. The hope is that image editing will see a dramatic increase in speed. Users must be able to edit multiple

files and objects simultaneously because they will be using the software to mix both sounds and images.

3.4 Logical Database Requirements

There will be no database used in the system.

3.5 Design Constraints

The only tangible design constraint is that CPython be used and PyGame be used for editing images and audio.

3.5.1 Standards Compliance

All code should be commented using the standard PyDoc commenting procedures. If the programmer is unfamiliar with that, please check
http://docs.python.org/tut/node6.html#SECTION006760000000000000000

3.6 Software System Attributes

3.6.1 Reliability

The system shall be using the underlying CPython language in order to run, therefore it cannot be more reliable than CPython itself. Having said this, if a developer is using media.py and errors are generated, all the errors should be passed to the developer, where they are allowed to handle them appropriately.

3.6.2 Availability

If the program does crash, the user should be able to restart immediately with only a loss of what they had typed since the last time they saved. The user is responsible for minimizing the effect of a computer crash by saving frequently.

3.6.3 Security

Although the system is not critical, and will not be used for network connections, it is still important to protect the user from local attacks. All input will be checked to make sure it will not cause buffer overflows, or add to system instability.

3.6.4 Maintainability

Maintainability is very important for this system because another group will be maintaining it. Therefore, it is very important that all programmers comment their code anytime something is unclear. Python makes indentation important, therefore the code is guaranteed to be more readable. At the beginning of each function there shall be PyDoc comments explaining what the function does and what input and output is expected.

3.6.5 Portability

1. The system must be able to run on Windows systems.
2. The system must be able to run on Macintosh systems.
3. The system must be able to run on Unix systems.
4. The code should be readable, well commented, and maintainable.
5. The system must be written in CPython.
6. The GUI must be implemented in an existing library, specifically wxPython.

3.7 Non-functional Requirement

All tests that run on the system shall be timed using the "time" function in Mac OS X, and a stopwatch in Windows. The tests will also be times using JES on both systems and their comparisons will be compiled for the customer.

4. Change Management Process

Additional requirements should be submitted to the group via email or in person. The group will then come to a consensus on the additional requirements, and the appropriate changes will be made. Mark Guzdial can make requirement request for the project, but no later than October 12. The team will evaluate the request by amount of resources it will take to implement the request. A team member may also deny the request if the requirement will impede speed increases, and the member has found a faster way to implement the same functionality.

5. Document Approvals

Andrew Nagel _____

Dustin Roberts _____

John Burton _____

Sam Gawthrop _____

Mark Guzdial _____

6. Supporting Information

Supporting Documentation:

www.python.org
www.pygame.org
www.wxwindows.org