

Cool Team Name

<http://walked.edwardpultar.com>

Software Design Specification Outline

Version: CS 4500, University of Utah
Spring 2005

1. Introduction

1.1 Purpose of this document

This SRS describes the function and performance requirements allocated to the WalkEd system. The WalkEd system is a multi-agent pedestrian modeling system for urban simulation.

1.2 Scope of the development project

The product to be developed is a program to allow users to create simulations of urban environments and realistically model pedestrian choreography within that environment. We may be limited as far as simulation size given restraints of computing power. There may also be restraints in graphic processing power and the level of choreography detail we want to implement. We hope to be distinct in our program by taking social science and geographical aspects into account more than previous attempts. Also we hope it can be used for urban planning, development, and academic model testing.

1.3 Definitions, acronyms, and abbreviations

Cal3D: Character Animation Library 3D

CEL: Crystal Entity Layer

CES: Complex Emergent Systems

CS: Crystal Space 3D

MAS: Multi-Agent System

1.4 References

SRS and references within the SRS. [<http://walked.edwardpultar.com/project/srs.pdf>]

1.5 Overview of document

The remainder of this document is organized around a template provided by Prof. Henderson. In it one will find descriptions of our system architecture and components.

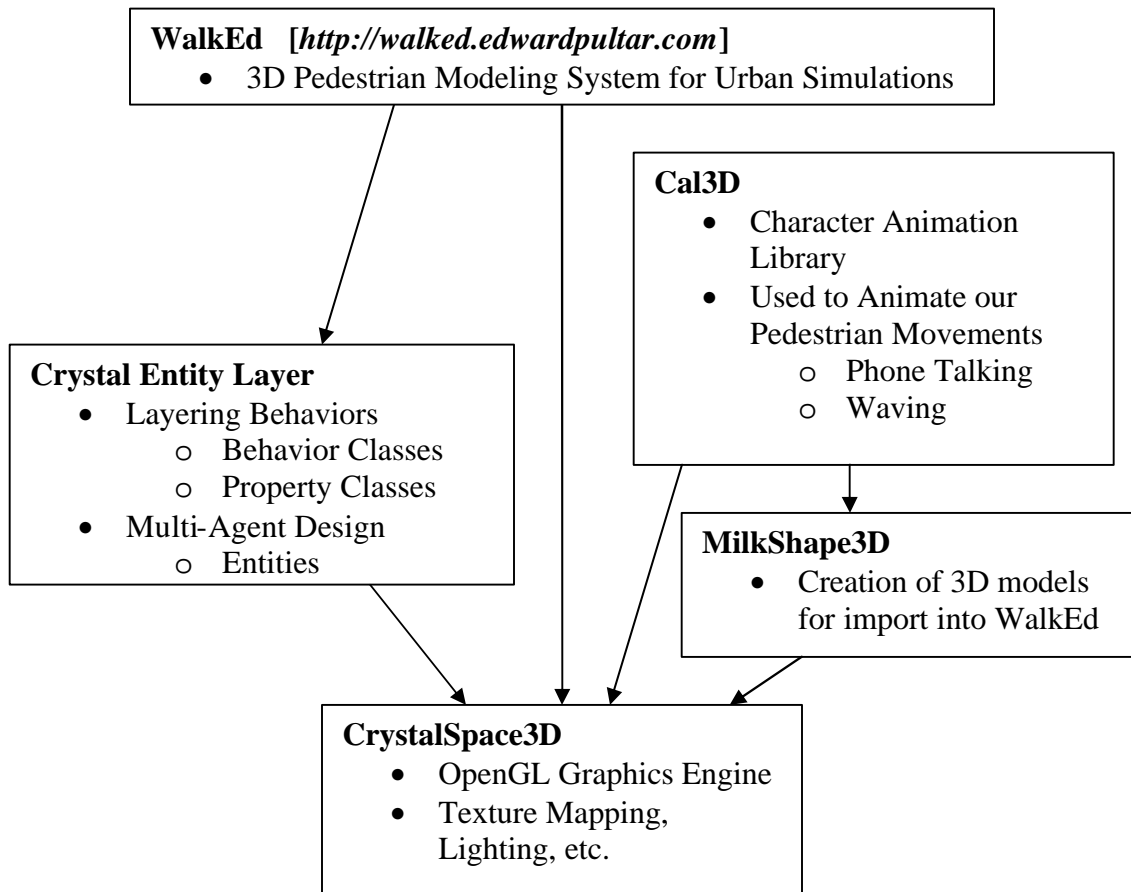
2. System architecture description

NOTE: This section is the main focus in version 1.0 of the SDS, the high level design. The reader should come away with a good view of exactly how your solution is to be organized.

2.1 Overview of modules / components

This subsection will introduce the various components and subsystems.

2.2 Structure and relationships



2.3 User interface issues

- **Academic Presenter** – The Academic Presenter uses many of the interactive features of the WalkEd system. Time rate controls are used, via a graphical interface such as a slider or stop button, which are also accessible via the system console. In addition to this, adding and removal of agents into a working system is achieved by mouse interfaces or console commands, and trimming of fine personality details of an agent are achieved during creation through a menu system. This menu can also be accessed for an existing agent via a mouse click, which allows for direct mutation of run-time agents.
- **Architect** – The Architect has an in depth set of interactions with the WalkEd system. In addition to run-time alterations, the main focus of work for the Architect goes into slight alterations of simulation parameters, such as desirability and agent details, paired with a great volume of simulations. These can be achieved through a batch-job interface provided through WalkEd.
- **Urban Planner** – The Urban Planner’s interaction the WalkEd system is slight. At a prearranged time, a simulation is crafted with certain agents, and few if any desirability entities attached to the surrounding environment. This initial setup is done during a previous run session through a semi-intuitive interface involving region-selection with the mouse, and click or console interactions on the selected region to activate an entity, type it (as an agent, desirability entity, etc) and apply basic features (movement rates, desirability levels, etc. The simulation is for purely demonstratory purposes, and as such no run-time interaction of any kind is required during the demonstration itself, except for loading of the preconfigured simulation parameters.

2.4 User interface specification

The user interface is concise and functional. Only key factors that the user needs to tweak are part of the interface. We plan to have the user be able to change pedestrian agents of the simulation by doing things such as adding and deleting agents. We also plan to include a pause button allowing the user to pause the simulation for inspection. Also we hope to have multiple camera views like first and third person available for the user to select.

3. Detailed description of components

3.1 Component template description

We will be using the following template to describe our components:

Identification	The unique name for the component and the location of the component in the system.
Type	A module, a subprogram, a data file, a control procedure, a class, etc
Purpose	Function and performance requirements implemented by the design

	component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements.
Function	What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.
Subordinates	The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.
Dependencies	How the component's function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.
Interfaces	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here.
Resources	A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services.
Processing	The full description of the functions presented in the <i>Function</i> subsection. Pseudocode can be used to document algorithms, equations, and logic.
Data	For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary.

3.2 Crystal Space Component (or Class or Function ...)

Identification	Crystal Space 3D
Type	3D OpenGL Graphics Engine
Purpose	The reason we have this component is so we do not have to focus on the graphics but rather AI and pedestrian choreography.
Function	Does Graphics operations such as lighting and texture mapping. Uses OpenGL to create good 3D worlds that our pedestrian entities can

	inhabit. Produces the 3D visualizations for WalkEd simulations.
Subordinates	The component is composed of C++ code and the internal structure can be referred to in the API at http://www.crystalspace3d.org .
Dependencies	This is the base component that all other components rely on for graphic display. Crystal Entity Layer is built on top of Crystal Space to allow us access to entities and their behaviors. Crystal Space is the glue that puts all the other components together that we rely on for visualization.
Interfaces	Crystal Space has consoles for input and output for the system. There also are 2D consoles that can be implemented into the system for GUI buttons and widgets.
Resources	This component requires the resources of a full PC with processor, memory, hard drive, and graphics card. The external resources needed are a keyboard, mouse, and monitor. It also requires the software of a C++ compiler in order to compile the code.
Processing	There are too many functions, etc. within CrystalSpace3D to list here, see the API at http://www.crystalspace3d.org for a full list.
Data	For the large amount of internal data contained within CrystalSpace3D reference the API found at http://www.crystalspace3d.org .

3.3 Crystal Entity Layer Component

Identification	Crystal Entity Layer
Type	Entity-Based Tool for Crystal Space 3D
Purpose	Allows us to divide the 3D world up into entities that enables multi-agent programming.
Function	Creates entities that have different layers and behaviors made for each entity type. Can use human actor entities as pedestrians.
Subordinates	The component is composed of C++ code and the internal structure can be referred to in the documentation at http://cel.sourceforge.net/ .
Dependencies	Our WalkEd system depends on CEL for letting us do agent-based programming and further making our Crystal Space programming more useful. Crystal Entity Layer depends on Crystal Space in order to work.
Interfaces	Since CEL is built on Crystal Space it has consoles for input and output for the system. There also are 2D consoles that can be implemented into the system for GUI buttons and widgets.
Resources	This component requires the resources of a full PC with processor, memory, hard drive, and graphics card. The external resources needed are a keyboard, mouse, and monitor. It also requires the software of a

	C++ compiler in order to compile the code.
Processing	There are too many functions, etc. within CEL to list here, see the API at http://cel.sourceforge.net/ for a full list.
Data	For the large amount of internal data contained within CEL reference the documentation found at http://cel.sourceforge.net/ .

3.4 MilkShape3D Component

Identification	MilkShape3D
Type	3D modeling tool.
Purpose	Graphical tool used in the creation and modification of 3D models.
Function	This program is used for creating models to use in the WalkEd system. The inputs processed are files and user input through peripherals. Outputs are files containing the various models created or modified.
Subordinates	This application is not open source; as such, the internal structure is unknown. For our purposes only the black box features are necessary.
Dependencies	This application does not depend on any other components.
Interfaces	The interface is predefined by its creator, chUmbaLum sOft. It interfaces with Cal3D and Crystal Space 3D by model files for rendering during simulation.
Resources	This component requires the resources of a full PC with processor, memory, hard drive, and graphics card. The external resources needed are a keyboard, mouse, and monitor.
Processing	There are too many functions, etc. within MilkShape3D to list here, see http://www.swissquake.ch/chumbalum-soft/ for a full list.
Data	For the large amount of internal data contained within MilkShape3D reference the documentation found at http://www.swissquake.ch/chumbalum-soft/ .

3.5 Cal3D Component

Identification	Cal3D
Type	Character Animation Library
Purpose	To animate characters using C++.
Function	This component's functionality is to animate characters that we create through MilkShape3D using Crystal Space code.

Subordinates	Imports models from various 3D modeling programs. It then applies animation techniques to the models.
Dependencies	This application depends on models but can stand alone even though that does not do much without character models.
Interfaces	This interfaces with Crystal Space which we will use through CEL for functionality.
Resources	This component requires the resources of a full PC with processor, memory, hard drive, and graphics card. The external resources needed are a keyboard, mouse, and monitor. For software it requires a C++ compiler to compile the code
Processing	There are too many functions, etc. within Cal3D to list here, see http://cal3d.sourceforge.net/ for a full list.
Data	For the large amount of internal data contained within Cal3D reference the documentation found at http://cal3d.sourceforge.net/ .

4.0 Reuse and relationships to other products

Reuse is playing a role in our product design by eliminating some of the graphics programming overhead. By using the CrystalSpace3D engine (<http://www.crystalspace3d.org>) we are able to focus more on our pedestrian choreography rather than details about texture mapping, lighting, etc. This also makes the project more feasible for the amount of time we have. This reuse of available open-source code is playing a role in our product implementation by allowing us to put more effort on our agent AI development and realistic behavior. These changes were very welcomed as we do not have the time or want to create a whole 3d graphics engine so now we are able to just implement our pedestrians.

5.0 Design decisions and tradeoffs

We decided to use this design of making use of existing code for the 3D graphics because of the time restraints of our project. We also are not so familiar with graphics that we would want to write our own engine, but we are very interested in creating these pedestrian simulations taking social science and geography factors into account. We also wanted to do this project in 3D so that it would be the most appealing to look at for potential customers.

6.0 Pseudocode for components

N/A

7.0 User Manual

The official user manual is distributed with the WalkEd system software. The README file also includes information a user may want to know for installing, running, and maintaining the system. As the program further develops more features the manual will be updated to reflect those changes.

Page prepared by Vicki L. Almstrum. Department of Computer Sciences at UT Austin.
Modified by Thomas C. Henderson. School of Computing, Univ of Utah