Shale

Design

May 3, 2009

Paul Gerhardt

Amanda Orin

Tomas Ramirez

Jessa Rothenberg

Kaiti Trimble

CSCI 4308-4318. Software Engineering Project 1 & 2

Department of Computer Science

University of Colorado at Boulder

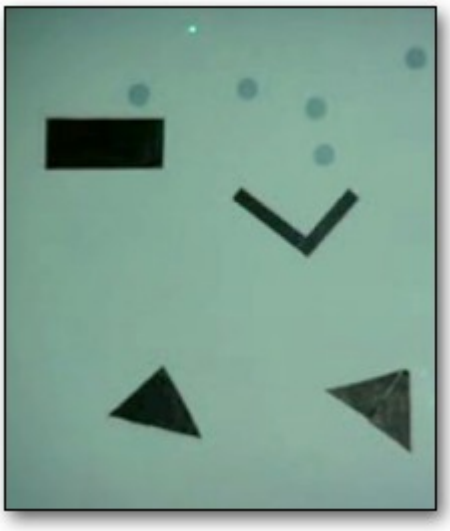
2008-2009

Sponsored By

CU Craft Technology Group

PROJECT PROPOSAL

The purpose of the project (Shale) is to construct a system in which physical and virtual objects interact. A previous implementation of this idea, Laser Ball, combined virtual bouncing balls with physical pieces of cardboard.



The image to the left shows what the Laser Ball system looks like. A virtual screen is created by projecting light onto a wall. Balls are created by briefly shining a laser pointer onto the screen. The balls fall down, and when they collide with the black pieces cardboard shapes, they bounce like a real ball bouncing off of a hard surface.

A major requirement of this new project is that the physical components of the system be dynamic in some way, for instance, a box might flash a light when it is hit, or a seesaw moves when a ball falls on its raised end. In addition to having a system with dynamic physical components, the project sponsor hopes to have a system that is fun-to-use, as it may have entertainment and educational applications in the future.

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

CU Craft Technology Group is a part of the Center for Lifelong Learning and Design at the University of Colorado specializing in the integration of computation and craft materials to produce mathematical or educational toys and activities for children. The group consists of a small number of faculty members and students working on the Boulder Campus of the University.

Previously, CU Craft Technology collaborated with a team of undergraduate software engineers to develop a project entitled Laser Ball. This program created a combination of virtual and physical elements through the use of digital image recognition. Project Shale is an expansion upon this project, and will include wireless communication with the physical objects, in order to enable more advanced interactions such as movements, lights, and sound.

A conceptual diagram of the overall system is presented in Figure 1. The figure shows the software to be implemented, Shale (Levers Shadows & Wheels). Shale receives input from the web camera and outputs data to the projector, which projects virtual objects (e.g. falling spheres) onto a whiteboard or blank wall. Affixed to the wall are mechanical/physical objects and the beam of a green laser pointer, which are seen by the web camera, and interpreted by digital image processing software within Shale. Shale then combines the web camera's input and correlates the locations of the virtual objects to create an interactive environment. Shale will also interact with the physical objects through use of a wireless transmitter. It will send signals to these physical objects when they interact with virtual objects, triggering movement (through motors), lights (through LEDs), or sounds (through speakers).

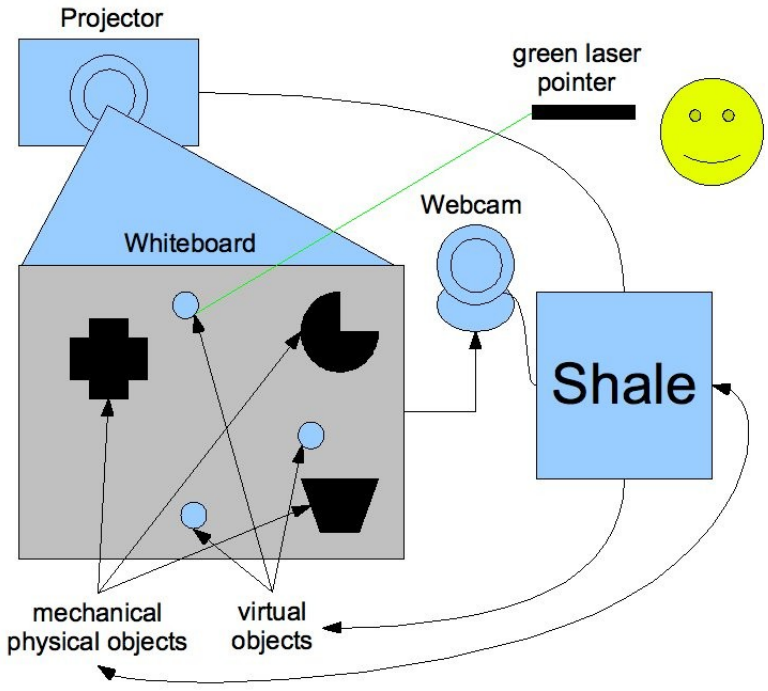


Figure 1: *Conceptual Overview of the Shale System*

This document defines the current design of Shale, which is intended to satisfy the requirements found in *Shale Requirements*. This design will be updated and expanded as it evolves over the course of the project.

There are two primary aspects of the design of Shale: user interface design and software architecture. Each of these design aspects is described below, followed by a list of useful references.

2 USER INTERFACE

Shale presents two graphical user interfaces: one for users and one for administrators. Users interact with Shale by directing a laser pointer onto a virtual window that is projected on a flat surface by a projector. The flat surface is referred to as the *Stage*, and the users' interface is referred to as the *Standard User Interface*. The laser pointer image is recorded by a web camera, and its location determines user input. Administrators interact with Shale through a standard operating system window displayed on the computer's monitor. The administrators' interface is referred to as the *Administrative Interface*.

2.1 Standard User Interface

Shale provides the normal, or "standard", user with a graphical user interface, through a virtual window that is projected on the *Stage*. This window, depicted in Figure 2 below, is composed of two major regions: *Current View* and *Menu Buttons*.

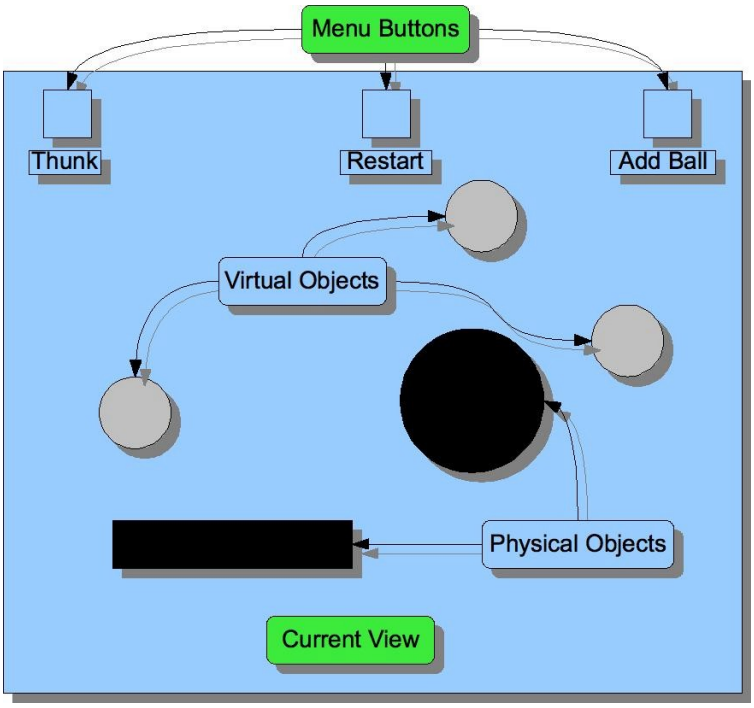


Figure 2: *The Shale Standard User Interface*

Each of these regions is discussed briefly below.

2.1.1 Stage

The external, flat surface upon which images are projected is referred to as the *Stage*. The *Stage* is the place where users shine the laser pointer. Users may attach real, physical objects to the *Stage*. While the program is running, virtual images, such as balls, will be projected upon the Stage, and their activity will be influenced by the physical objects that are on the *Stage*. For example, virtual balls could appear to bounce off of the physical objects. Some of the physical objects can be programmed to respond to signals from the Shale program. More details about virtual objects and physical objects are given below.

2.1.1.1 Virtual Objects

The objects that are being displayed by the projector onto the *Stage* but cannot be physically manipulated by the user are the *Virtual Objects*. These objects react to the presence of *Physical Objects* and other *Virtual Objects* and are under the influence of artificial gravity. For instance, one basic *Virtual Object* currently used as the default is a ball that bounces off other balls or *Physical Objects*.

2.1.1.2 Physical Objects

The objects on the *Stage* that can be physically manipulated by the user and react to the presence of *Virtual Objects* are the *Physical Objects*. For instance, if a ball collides with a *Physical Object* like a seesaw, the *Physical Object* will rotate in reaction to the collision as the ball bounces off it, so the user sees the collision as if it were two actual objects reacting on the *Stage*.

2.1.2 Current View

In essence, *Current View* consists of the images displayed on the *Stage*. The *Current View* is the region of the interface in which the user can view the *Virtual Objects* currently being projected (e.g., silver balls in Figure 2) in addition to the *Physical Objects*. If the user shines the laser pointer in the *Current View* region for two seconds, then a *Virtual Object* will appear in that location. The *Virtual Object* will immediately begin moving and interacting. Currently the only kinds of *Virtual Objects* are silver balls that fall and bounce off of *Physical Objects*, but in the future, more *Virtual Objects* will be possible, and a new interface feature will be introduced for creating *Virtual Objects*. When the *Current View* first appears, no *Virtual Objects* will be on the screen. Once the first *Virtual Object* is created, it will begin moving and interacting with *Physical Objects*.

2.1.3 Menu Buttons

The *Menu Buttons* are designated by labeled boxes in the upper region of the screen. The *Menu Buttons*, **Thunk**, **Restart**, and **Add a Ball**, are described below. *Menu Buttons* are selected by directing the laser pointer at them for at least two seconds. While a *Menu Buttons* is selected, a change in color of that button's box will appear.

Menu Buttons		
	Thunk	The **Thunk** button acts like a fist that hits a side of the application window – it causes the *Virtual Objects* to move and interact with other *Virtual Objects* and *Physical Objects*.
	Restart	The **Restart** button clears any existing *Virtual Objects* from the display, resets the state of all *Physical Objects* and plays Shale.
	Add a Ball	The **Add a Ball** button adds a new *Virtual Object* to the screen.

2.2 Administrative Interface

Shale provides the administrator with a graphical user interface through a window in the computer's operating system. This window, depicted below in Figure 3, is composed of *Current View*, *Status Bar*, and *Menu Bar* regions.

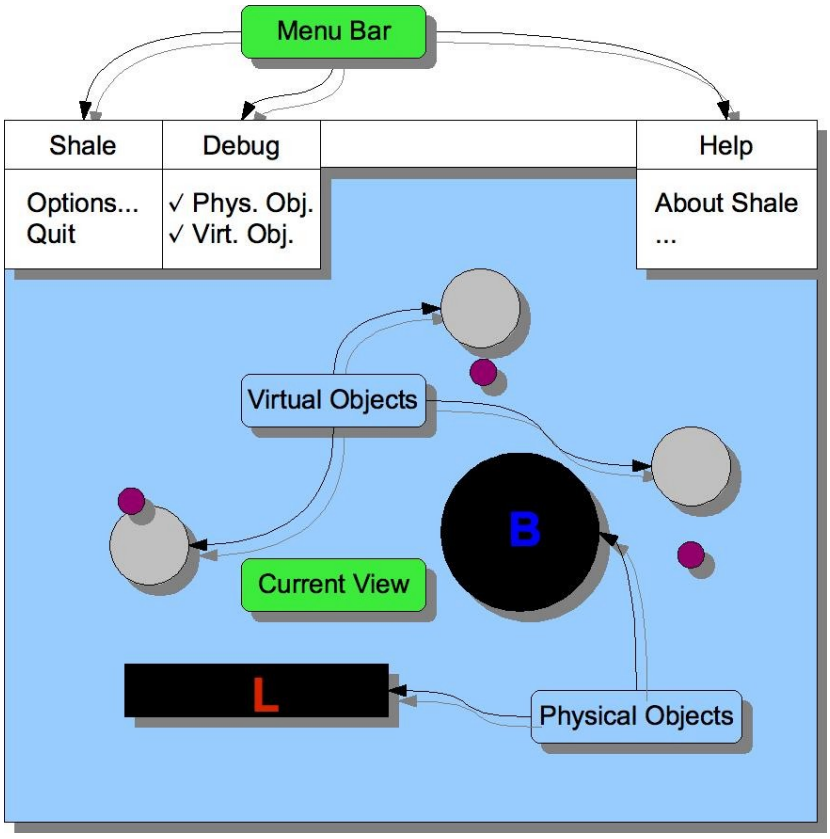


Figure 3: *The Shale Administrative Interface Window*

A brief description of each region is provided below.

2.2.1 Current View

The *Current View* is the region of the window in which the administrator can view the *Virtual Objects* currently being projected on the *Stage* (e.g., silver balls in Figure 3) as well as what is being recorded by the web camera (the physical objects on the *Stage*).

2.2.2 Menu Bar

The *Menu Bar* provides three pull-down menus: **Shale***,* **Mode**, and **Help**. The menu hierarchy is expanded below, with descriptions of each command.

Shale →		
	Options…	Opens a dialog box shown below in Figure 4.
	Quit	The **Quit** command exits the program.

Debug →		
	Virtual Objects →	Can be toggled to turn on/off debugging features for the virtual objects in the current view, such as a directional vector that indicates which way the object is currently moving.
	Physical Objects →	Can be toggled to turn on/off debugging features for the physical objects in the current view, such as their ID and their current status.

Help →		
	About Shale	The **About Shale** command opens a window that contains the Shale logo, any copyright information, the version number, and a link to Project Shale's website.
	(help topic 1)	No description at this time.
	(help topic 2)	No description at this time.
	…	

The *Options Dialog Box* (shown in Figure 4) lets the administrator adjust restrictions on the *Virtual Objects*. The administrator can set the maximum and minimum number of *Virtual Objects* shown on the screen at any one time. Ideally this will be portrayed using sliders, with a minimum value of zero and a maximum value based on performance constraints.

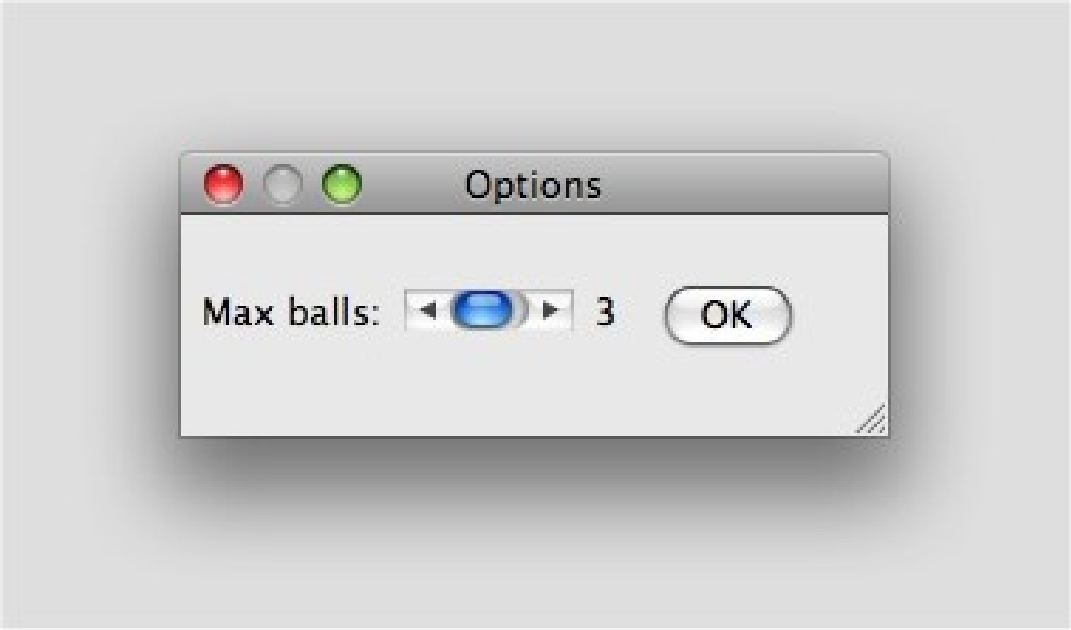


Figure 4: *Shale Options Dialog Box*

3 ARCHITECTURE

This section describes the architectural design of Shale. At the highest level, Shale is composed of six primary modules:

- *GUI* (Graphical User Interface)
- *Image Recognizer*
- *Objects*
- *Signal Token Pool*
- *Object Controller*
- *Collision Detector*

Figure 5 below shows the interactions among the modules.

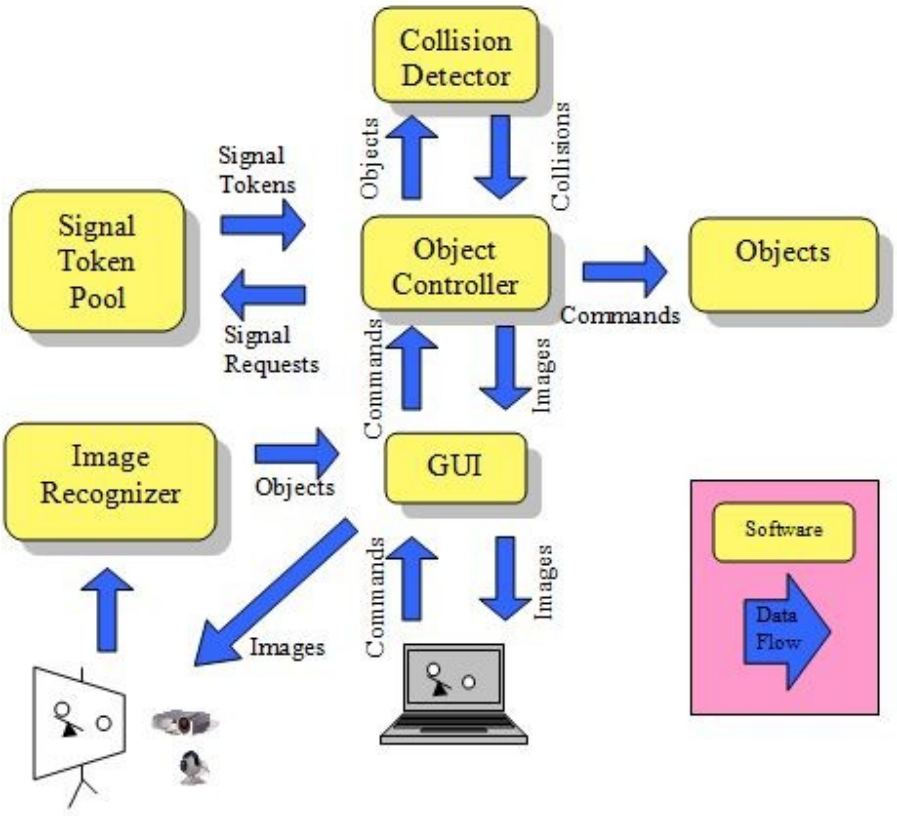


Figure 5: *High-Level Decomposition of the Shale System*

An overview of each of Shale's modules is provided below.

3.1 GUI

The GUI (graphical user interface) is responsible for displaying *Object* images and menus. The GUI is also responsible for handling command selections made in the *Administrative Interface* and the *Standard User Interface*. The GUI relies on input from the *Image Recognizer* to determine whether a command has been selected in the *Standard User Interface*.

3.2 Image Recognizer

The primary function of the *Image Recognizer* is to determine the outlines of all objects that are in the view of the web camera on the *Stage*, and determine the location of the laser pointer. The *Image Recognizer* creates an *Object* for each object that it recognizes in the web camera image data. The *Image Recognizer* also sends laser pointer and *Object* data to the GUI, which handles the data appropriately.

3.3 Objects

The *Objects* are abstract representations of both physical and virtual objects. There are two types of *Objects*: *Virtual Objects* and *Physical Objects*. The *Collision Detector* determines which objects have collided, and activates collision behavior of *Objects*. The *Object Controller* activates movement behavior of each *Object*.

3.3.1 Virtual Objects

Virtual Objects are objects that only exist as a light-image projected by a projector. *Virtual Objects* can collide and move.

3.3.2 Physical Objects

Physical Objects are objects that exist in the real world and are recognized by the *Image Recognizer*. *Physical Objects* can collide and move. *Physical Objects* will perform an action upon collision which will be triggered by the object controller. For *Physical Objects*, this involves activating the *Signal Token* that the *Physical Object* has subscribed to.

3.4 Signal Token Pool

Signal Tokens are application level encoded network packets containing instructions specific to each *Physical Object*. These instructions may include signals to start and stop stepper motors, chirp, or otherwise interact with a *Physical Object's* hardware.

The *Signal Token Pool* contains and issues all *Signal Tokens*. The *Signal Token Pool* is responsible for ensuring that all *Signal Tokens* have a unique *Signal*. *Physical Objects* are assigned a unique *Signal Token* that is requested from the *Signal Token Pool*. The *Signal Token Pool* is also responsible for interfacing with the communication hardware for passing *Signal Tokens* to the *Physical Objects* and for receiving movement commands from the *Object Controller*.

3.5 Object Controller

The *Object Controller* is responsible for controlling all *Objects*. It contains the collection of all *Objects*. *Physical Objects* are added to the *Object Controller* by the *Image Recognizer*. *Virtual Objects* are generated by the *Object Controller*, but the number of *Virtual Objects* is based on command data from the GUI. The *Object Controller* makes use of a *Collision Detector* that evokes the collision behavior of the *Objects*.

3.6 Collision Detector

The *Collision Detector* is responsible for detecting collisions between all *Objects* in the *Object Controller*. Collision detection is performed by measuring the distance between the center of an *Object* and every other *Object* on stage, for each *Object*. If the *Object's* size is greater than the distance to a nearby *Object*, and both *Objects* are *Virtual* than new velocity vectors are calculated for the *Virtual Objects*. If one of the *Objects* is *Virtual* and the other is *Physical*; a new velocity vector is calculated for the *Virtual Object*, an appropriate reaction is determined for the *Physical Object*, and a movement command is issued to the *Object Controller* to request from the *Signal Token Pool*. Collisions between two *Physical Objects* are ignored.

4 SUMMARY

This document describes the current Shale design. It first provided a description of the user interface, both an administrative user interface for administrative operation, and a standard user interface for interaction from the projected image. The high-level architecture decomposed the software into its six primary components – *GUI*, *Image Recognizer*, *Collision Detector, Signal Token Pool, Objects* and *Object Controller*.