6.170
Preliminary Design Document

SE64 Gadgix Presents: Gizmoball

Austin Chu (eefi)
Benjamin Gleitzman (gleitz)
John Hawkinson (jhawk)
Bill Magnuson (billmag)

April 18, 2007
# Contents

1. Overview ........................................................................................................................................... 1

2. Design .................................................................................................................................................. 1
   - Module Relationships ......................................................................................................................... 2
   - Alternatives Considered ..................................................................................................................... 3

3. Graphical User Interface ..................................................................................................................... 4
   - Location of Controls ........................................................................................................................... 4
   - Changes in the Game Display ........................................................................................................... 4
   - Gizmo Placement ............................................................................................................................... 5
   - Connections ...................................................................................................................................... 5
   - Menus ............................................................................................................................................... 5
   - Stop vs. Pause .................................................................................................................................. 6

4. Project Plan .......................................................................................................................................... 6
   - Group Formation April 6 ................................................................................................................... 6
   - Final project preliminary design April 18 .......................................................................................... 6
   - Milestone 1 (Specifications) April 22 ................................................................................................. 7
   - Milestone 2 (Static board loading) April 26 ....................................................................................... 7
   - Milestone 3 (Preliminary release) May 1 ............................................................................................. 7
   - Milestone 4 (Building mode) May 8 .................................................................................................... 7
   - Final release May 14 .......................................................................................................................... 7
   - Work Allocation ................................................................................................................................ 7
   - Project Chart ..................................................................................................................................... 8

Appendix A ............................................................................................................................................. 9
   - API documentation for Gizmoball ....................................................................................................... 9

Appendix B ........................................................................................................................................... 11
   - Gizmoball GUI Images ...................................................................................................................... 11
1. Overview

Gizmoball is a three dimensional version of pinball, an arcade game in which the object is to keep a ball moving around in the game, without falling off the bottom of the playing area. The player controls a set of flippers that can bat at the ball as it falls.

The advantage of Gizmoball over a traditional pinball machine is that Gizmoball provides a mode separate from the normal game-playing mode in which users may construct their own machine layouts by placing gizmos (such as bumpers, flippers, and absorbers) on the playing field. Gizmoball also allows users to save their layouts and load saved layouts in a standard XML file format.

2. Design

The core of the Gadgix Gizmoball system is the emulation of the virtual world of the Gizmoball playing area. Each of the objects – including SquareBumpers, CircleBumpers, TriangleBumpers, LeftFlippers, RightFlippers, Absorbers, Walls, and Balls – are first-order objects that will track their own locations, velocities, orientations, and other state. All of these items will implement the Gizmo interface. There also exists a Board object, which will encapsulate properties of the world (like gravity and friction) and keep track of all the Gizmos that exist.

The system uses a separate set of objects to display the virtual world for the user. Each Gizmo object will have an associated GizmoView object that will know how to appropriately render it to the screen. In addition, a Display object will keep track of all the GizmoView objects that exist.

There are a number of other classes that are required in the system. The Actions class dispatches user inputs to Gizmos (e.g., for keyboard-triggered actions). The Actions class gets its inputs from an input-method-specific Input class, which can also handle other user inputs, such as changing modes and starting and stopping the simulation. The Clock class provides an abstraction for time in the virtual world. The BuildMode class oversees the building mode of Gizmoball. Finally, the BoardBuilderXML class processes Gizmoball level files.
Module Relationships

At the highest level, the design of the Gadgix Gizmoball system is organized around the traditional model-view-controller design pattern. The Board and Gizmos provide the model of the world being emulated. The Display and GizmoViews provide the user a view of that model. Managing this view and relaying user input to the model are the Main, BuildMode, Input, and Actions modules.
As part of the model-view-controller paradigm, the system tries to decouple the model – the Board and the Gizmos – from any modules dealing with display as much as possible. The relationship between Gizmo and GizmoView is modeled after the observer design pattern, with GizmoView observing Gizmo; thus, the dependency of Gizmo on GizmoView is only a weak one. Such decoupling will enable easily adding new display modes, such as an ASCII art display.

The desire to enable additions of possible future display modes also motivates the decoupling of the mode-specific Input module from the generic Actions module. The Actions module will be able to trigger Gizmo actions generically based on events generated by the Input module. Different Input modules can then be written to generate events based on input read from a GUI framework or from input read from a console.

**Alternatives Considered**

For the decomposition of the basic model, the organization of the system around the Gizmos as first-order objects with an overseeing Board object seemed to be the most natural abstraction. There was some thought given to having a separate Physics module that would allow easy tweaking of physics parameters, e.g. special gravity, oil, etc., but it was determined that folding those constants into the Board module made the most intuitive sense.

We strongly debated the decomposition of how to generate a display. The first proposal had each Gizmo itself in charge of its own display, with each class implementing the Gizmo interface for example having a drawGl(), a drawAscii(), and whatever other draw*() methods required. We decided this was suboptimal because addition of new display modes would require modifying existing modules, and the size of each Gizmo module would become unnecessarily larger as display modes were added.

A second proposal had a single display module that would have a draw*() method for each Gizmo. We determined that this was suboptimal because such a module would become very large, and having all of the draw*() methods for all Gizmos within one class would mean modifying individual draw*() methods concurrently would be more difficult.

Thus, in the end, we decided on having a separate GizmoView per Gizmo per display mode to decouple display code from modelling code and to modularize the display code into more easily concurrently modifiable files.

A second debate that we had over the display decomposition was whether to have a Display class that kept a list of GizmoViews parallel to the list of Gizmos kept by Board. Having a Display class had the advantage that Board and Gizmo would have as little to do with displaying themselves as possible, but having two separate lists of GizmoViews and Gizmos ran the risk of their getting out of sync. Having no display class meant that redraw requests would have to go
through Board, slightly breaking the model-view separation, but not having a separate list of GizmoViews would mean having only one canonical list of Gizmos (with their Views accessible through the Gizmos since they had already been registered with the Gizmos via the observer pattern). In the end, our team deadlocked on this decision and went with the former alternative based on our arbitration procedure.

Finally, we are still debating whether the Actions module should track triggers generated by Gizmos in addition to triggers generated by user input. A possibility would be to have Gizmos able to listen to each other for triggers generated by other Gizmos.

3. Graphical User Interface
The graphical user interface provides a user with the ability to control a large majority of all available public operations in Gizmoball.

Location of Controls
The first design task was to allocate space in the window for a toolbar that provides a control mechanism. In order to provide an analog with the real world, we decided to place the controls on a vertical bar to one side of the game display. This is because pinball is a vertically oriented game and we wanted to maximize available vertical space by orienting minimal UI across the top or bottom of the window.

The next requirement is to accommodate user input in both building mode and running mode. In order to make clear distinction between the two modes, we decided to have the entire toolbar be replaced when a user toggles between the modes. Furthermore, the color scheme for each set of tools is different. In order to switch between the two modes, the user must click on a button placed at the bottom of each toolbar. If the game is paused or the board has been saved then the switch will happen without prompting the user. In the event of an accidental click, the switch can be easily reversed without consequence. If the game is not in a safe transition state, the user will be prompted with a warning to fix the situation. Once the user stops play of the game, or saves the board if in building mode, they will then be able to switch between modes without a prompt.

Changes in the Game Display
Running mode and building mode have a variety of different view requirements in the game display which must change when the user switches modes. The building mode requires the addition of a number of elements to help the user create a new board or see the details associated with a board. Since the UI will allow for a game to be going on (but paused) while a player switches to building mode it is important that the current ball be shown as well as the starting location and velocity vector of the ball for the board at reset. The starting location will
be represented by another graphic of a ball that distinctly separates its appearance from the active ball.

Upon entering building mode a grid is superimposed on the existing board and gizmos are forced to snap onto discrete grid coordinates. Another graphical change is the addition of details about the different gizmos on screen while in building mode. Some of the details (like connections or coordinates) will be viewable in the additional information section of the right side control panel, and other information will be drawn into the game display. Some of the drawn information includes velocity vectors (for the ball, or perhaps future gizmos that can move), specific descriptors for gizmos such as the direction of a flipper and proximity boundaries around the gizmos.

**Gizmo Placement**
Placing gizmos was one aspect of the UI that we aimed to be completely unambiguous, but still graphically appealing to the user. The control scheme is as follows: a user will select a gadget to add and upon clicking it will be placed in an open space on the board near the center. The gadget is now in the playing field. In order to move it to the desired location the user will be required to click on the gadget, they will receive a visual confirmation of the selection in two locations; first the gadget specific information will be filled in and second the gadget itself will undergo a slight color change. The user will then click on the directional pad below the gizmo palette to move the gizmo one grid location at a time. Rotation is done with the two arrows on the outside and represent a 90 degree twist either clockwise or counterclockwise per click. To move items in the z direction the forward and backward text label buttons are used, they represent a move towards the user and away from the user respectively. These were chosen to be text label buttons instead of icons because no obvious icon exists to represent z axis movement.

**Connections**
Connections are handled with the “Connect” button on the sidebar. The user will select a gizmo and then press Connect. The gadget specific information will be updated to list any existing connections and the user will then be required to either press a key to set a trigger, or select another gizmo with the mouse to set an action chain.

**Menus**
The traditional menus in Gizmoball provide a limited functionality set by only providing the menu options that a user would traditionally associate with a menubar. This means things like resetting the game and managing the board (such save, save as and loading) are available in the file menu, but other commands like adding gadgets are only available through the sidebar. This was done for two reasons. The first is to keep the menus uncluttered and the second is to better separate the two different modes. Since a user would only be using a large amount of
the feature set in one mode, but not the other, we decided that these non “cross-mode” operations should not be included in the menubar.

The help menu is also very predictable in its behavior, providing help contents and an about page. The help contents will link to the program's documentation and provide the user with a variety of information on the workings of the system as well as descriptions of all available gizmos.

**Stop vs. Pause**
The running mode UI provides two graphical buttons for play and pause as well as a text labeled button to reset the board to its defined starting state. This three option choice was opted for over a “start/stop” set of options because the functionality of stop is under defined. It would be unclear to the user whether stop is going to stop the action on the board, but save its state, or if stop is going to stop everything and reset to the beginning.

Graphics for play and pause were both used because they are extremely common and well known graphics which will immediately convey their purpose with a quick glance. Reset is a text label button because no clear symbol exists for reset. It is placed below the grouped play and pause buttons to further visually separate the ideas of maintaining the state of the game and starting a new one.

### 4. Project Plan

The project schedule and workload divisions are achieved through the project plan, which includes milestones for the team and allocation of tasks for particular members. In addition to the preliminary design and final deadline, four individual milestones have been enumerated for the group.

**Milestones**

**Group Formation**

- Group members selected (all)
- Team name chosen (all)

**Final project preliminary design**

- Preliminary design document (all)
  - System architecture design (all)
  - API design (jhawk)
  - GUI prototype (billmag)
Milestone 1 (Specifications) April 22

- Existing specifications for all project modules (all)
- Determine division of labor for module design (all)

Milestone 2 (Static board loading) April 26

- At least skeleton classes existing for all modules (all)
- Gizmo classes written to the point that gizmos can at least exist
- GizmoView classes that can display gizmos on the screen
- Ability to load files in the standard format

Milestone 3 (Preliminary release) May 1

- Handle progression of time
- Handle collisions
- Handle hooking absorber up to keyboard
- Handle hooking flippers up to keyboard
- Complete running mode that can run any board loaded from a file

Milestone 4 (Building mode) May 8

- Complete building mode implemented

Final release May 14

- Final design document
- Optional features

Work Allocation

For original module specifications, each team member will create specifications for the following classes, to be completed by April 18, 2007.

<table>
<thead>
<tr>
<th>Class</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board, BoardBuilder</td>
<td>billmag</td>
</tr>
<tr>
<td>GizmoView Interface, Display</td>
<td>eefi</td>
</tr>
<tr>
<td>Main, Clock, Input Interface</td>
<td>gleitz</td>
</tr>
<tr>
<td>Gizmo Interface, Scorekeeper</td>
<td>jhawk</td>
</tr>
</tbody>
</table>

Figure 2 Module Responsibilities
Project Chart

A Gantt chart has been created to visualize project workflow through the final release.
Appendix A

API documentation for Gizmoball

Main class

- Main() "Start the ball rolling."

Clock class implements ModeInterface

- Start() Start the clock running.
- Stop() Stop the clock running. Can be resumed (re-Start()ed).
- Reset() Reset the stopped clock.

Board class

- private vars: list of objects, dimensions/size of board, modes, properties of world (gravity, friction, etc.)
- Create()
- Load(filename)
- AddGizmo()
- RemoveGizmo()

BoardBuilderXML class

Gizmo Interface

- Step() A time step has occurred
- Create() Create a gizmo
- Destroy() Destroy a gizmo
- Move() Move a gizmo
- Helptext() Return the help text associated with the gizmo.
  - Might also be used to generate docs at end of project?
- Proximate?() Am I close to something else?
- Collision() Am I colliding with something else?
- GetVelocity() Get velocity (only makes sense for ball)
- SetVelocity() ibid
- Action() action callback function
GizmoView Interface

Display interface

- Create(style) Where style is ASCII, GL, etc.

Input Interface

- Create(Style) Where style is ASCII, GL, etc.

BuildMode class implements ModelInterface

- Start() enter build mode
- Stop() exit build mode
- Reset() no-op

ScoreKeeper class

- AddToScore()
- GetScore()
- SetScore()

Actions class

- RegisterGizmo(gizmo, action) Register a gizmo as interested in an action.
- DispatchAction(action) Notify all registered gizmos that an action has taken place.
Appendix B

Gizmoball GUI Images

Gizmoball Run Mode
Gizmoball Build Mode
Gizmoball File Menu
Gizmoball Help Menu