Architecture Design

For Environment Model Building Tool (EMBT)

Version 1.1

Submitted in partial fulfillment of the requirements of the degree of MSE

Esteban Guillen
CIS 895 – MSE Project
Kansas State University
Table of Contents

1. Introduction ................................................................................................................. 5
2. Environment Model Builder .......................................................................................... 5
  2.1. Package View ........................................................................................................... 5
  2.2. Application Package ................................................................................................. 6
    2.2.1. Class Descriptions and Diagrams .......................................................... 6
    2.2.1.1. EMBApplication ................................................................................. 6
  2.3. Controller Package ................................................................................................. 6
    2.3.1. Class Descriptions and Diagrams ........................................................ 6
    2.3.1.1. EMBController ................................................................................... 6
    2.3.1.2. EMBBuildingSurfaceMouseHandler ........................................ 7
    2.3.1.3. EMBOBJECTPROPERTIESWINDOW ........................................ 7
  2.4. View Package .......................................................................................................... 8
    2.4.1. Class Descriptions and Diagrams ........................................................ 8
    2.4.1.1. EMBView ........................................................................................... 8
    2.4.1.2. EMBTHREEDIMENSIONALVIEW ............................................. 8
    2.4.1.3. EMBXMLVIEW ................................................................................. 9
    2.4.1.4. EMBDRAWINGVIEW ................................................................. 9
    2.4.1.5. EMBTERRAINPREVIEW ......................................................... 9
    2.4.1.6. EMBTERRAINVIEW ................................................................. 10
    2.4.1.7. EMBTERRAINVIEW ................................................................. 10
    2.4.1.8. EMBOBJECTPREVIEW ....................................................... 11
    2.4.1.9. EMBOBJECTVIEW ................................................................. 11
    2.4.1.10. EMBOBJECTVIEW .............................................................. 11
    2.4.1.11. EMBBUILDINGSURFACE .................................................. 12
  2.5. Model Package ....................................................................................................... 12
    2.5.1. Class Descriptions and Diagrams ........................................................ 13
    2.5.1.1. EMBMODEL .......................................................... 13
    2.5.1.2. EMBENVIRONMENT ......................................................... 13
    2.5.1.3. EMBOBJECT .......................................................... 13
    2.5.1.4. EMBBASICSHAPE .......................................................... 14
    2.5.1.5. EMBBOX .......................................................... 15
    2.5.1.6. EMBCONE .......................................................... 15
    2.5.1.7. EMBSPHERE ......................................................... 16
    2.5.1.8. EMBCYLINDER .......................................................... 16
    2.5.1.9. EMBTERRAIN .......................................................... 16
    2.5.1.10. EMOBJECTLIBRARY ............................................... 17
    2.5.1.11. EMBTERRAINLIBRARY .................................................. 17
  2.6. Sequence Diagrams ................................................................................................. 18
    2.6.1. Opening a Saved Environment ......................................................... 18
    2.6.2. Adding a Terrain to the Building Surface .................................. 18
    2.6.3. Adding an Object to the Building Surface ...................................... 19
3. Environment Object Builder ......................................................................................... 19
  3.1. Package View ......................................................................................................... 19
  3.2. Application Package ............................................................................................... 20
3.2.1. Class Descriptions and Diagrams .................................................. 20
  3.2.1.1. EOBApplication ........................................................................ 20

3.3. Controller Package ........................................................................ 21
  3.3.1. Class Descriptions and Diagrams .................................................. 21
    3.3.1.1. EOBController ....................................................................... 21
    3.3.1.2. EOBBoxPropertiesWindow ..................................................... 22
    3.3.1.3. EOBConObjectPropertiesWindow ......................................... 22
    3.3.1.4. EOBBoxPropertiesWindow ..................................................... 23
    3.3.1.5. EOBFrontMouseHandler ....................................................... 24
    3.3.1.6. EOBSideMouseHandler ......................................................... 24
    3.3.1.7. EOBSpherePropertiesWindow ............................................... 25
    3.3.1.8. EOBTopMouseHandler .......................................................... 25

3.3.1.9. EOBTopMouseHandler .......................................................... 25

3.4. View Package .................................................................................. 26
  3.4.1. Class Descriptions and Diagrams .................................................. 26
    3.4.1.1. EOBView ............................................................................... 26
    3.4.1.2. EOBThreeDimensionalView .................................................. 26
    3.4.1.3. EOBDrawingView .................................................................. 27
    3.4.1.4. EOBXMLView ....................................................................... 27
    3.4.1.5. EOBSideDrawingView .......................................................... 28
    3.4.1.6. EOBFrontDrawingView ......................................................... 28
    3.4.1.7. EOBTopDrawingView ........................................................... 28
    3.4.1.8. EOBObjectPreview ............................................................... 28
    3.4.1.9. EOBObjectFinder ................................................................. 29
    3.4.1.10. EOBObjectView ................................................................. 29

3.5. Model Package ............................................................................... 30
  3.5.1. Class Descriptions and Diagrams .................................................. 30
    3.5.1.1. EOBModel ............................................................................ 30
    3.5.1.2. EOBObject .......................................................................... 31
    3.5.1.3. EOBBasicShape .................................................................... 31
    3.5.1.4. EOBBox .............................................................................. 32
    3.5.1.5. EOBConObject ...................................................................... 33
    3.5.1.6. EOBSphere .......................................................................... 34
    3.5.1.7. EOBConObject ...................................................................... 35
    3.5.1.8. EOBObjectLibrary ............................................................. 35

3.6. Sequence Diagrams ....................................................................... 35
  3.6.1. Modifying and Moving A Box Shape .......................................... 36

4. Environment Terrain Builder ............................................................... 36
  4.1. Package View ............................................................................... 36
  4.2. Application Package .................................................................... 37
    4.2.1. Class Descriptions and Diagrams .............................................. 37
      4.2.1.1. ETBAApplication ............................................................... 37
  4.3. Controller Package .................................................................... 37
    4.3.1. Class Descriptions and Diagrams .............................................. 37
      4.3.1.1. ETBController ................................................................. 37
  4.4. View Package ............................................................................. 39
4.4.1. Class Descriptions and Diagrams ................................................................. 39
  4.4.1.1. ETBView .................................................................................................. 39
  4.4.1.2. ETBThreeDimensionalView ................................................................. 39
  4.4.1.3. ETBXMLView ......................................................................................... 40
  4.4.1.4. ETBDrawingView .................................................................................... 40
  4.4.1.5. ETBBuildingSurface ............................................................................... 40
  4.4.1.6. ETBTerrainPreview .................................................................................. 40
  4.4.1.7. ETBTerrainFinder .................................................................................... 41
  4.4.1.8. ETBTerrainView ....................................................................................... 41

4.5. Model Package ................................................................................................ 42
  4.5.1. Class Descriptions and Diagrams ............................................................. 42
    4.5.1.1. ETBModel .............................................................................................. 42
    4.5.1.2. ETBTerrain ............................................................................................ 43
    4.5.1.3. ETBTerrainLibrary ............................................................................... 43

4.6. Sequence Diagrams .......................................................................................... 44
    4.6.1. Modifying the Elevation of a Section of the Terrain ................................ 44

5. Formal Specification for the Environment Model Builder ..................................... 44
  5.1. USE Model ................................................................................................... 44
1. Introduction
   This document will provide brief descriptions and class diagrams of the applications and classes for the EMBT.

2. Environment Model Builder
   The Environment Model Builder is a graphical tool to create an environment from a terrain and objects. The tool will have a building surface to place the terrain and objects. The user will be able to move the objects to the desired location. There will also be a three dimensional view to observer what the terrain and objects look like in 3D. The following sections will describe the different packages of the Environment Model Builder in detail.

2.1. Package View
2.2. Application Package

2.2.1. Class Descriptions and Diagrams

2.2.1.1. EMBApplication
This class is just intended to have the main method for this program and create the EMBController and set it visible.

2.3. Controller Package

2.3.1. Class Descriptions and Diagrams

2.3.1.1. EMBController
This class is the main frame of the application. It will handle all the menu item actions. It is responsible for loading files, saving files to disk, and saving EMBEnvironments to the library.
2.3.1.2. EMBBuildingSurfaceMouseHandler

This class is responsible for handling mouse events for the building surface. In particular it will wait for mouse clicks and determine if one of the objects was clicked on. If an object is clicked on it will provide a properties window for that object.

2.3.1.3. EMBObjectPropertiesWindow

This class provides the properties window for an object. It will provide controls to move the object to a new location.
2.4. View Package

2.4.1. Class Descriptions and Diagrams

2.4.1.1. EMBView

This class is a container for the EMBThreeDimensionalView, EMBDrawingView, and EMBXMLView.

2.4.1.2. EMBThreeDimensionalView

This class will show the three dimensional view of the current EMBEnvironment. From this view the user will be able to view the EMBEnvironment from any angle.
2.4.1.3. EMBXMLView

This class is responsible for displaying the XML definition of the current EMBEnvironment. The XML will represent the contents that will be saved to disk.

```
EMBXMLView
  - createScene3D(): void
  - lightScene(): void
  - addBackground(): void
  - addControls(in : Canvas3D): void
  - initUserPosition(): void
  - clearView(): void
  - setVisible(in : boolean): void
  - updateView(Listener): void
  - addTerrain(in : BranchGroup): void
```

2.4.1.4. EMBDrawingView

This class is a container for the EMBBuildingSurface, EMBTerrainPreview, EMOBJECTPreview.

```
EMBDrawingView
  - builder: EMBBuildingSurface
  - objectPrev: EMOBJECTPreview
  - terrainPrev: EMBTerrainPreview
  - builderBorderPanel: JPanel
  - objectPrevBorderPanel: JPanel
  - terrainPrevBorderPanel: JPanel
  - previewPanel: JPanel
```

2.4.1.5. EMBTerrainPreview

This class is a container for the EMBTerrainFinder and EMBTerrainView. It will also add the currently selected EMBTerrain to the EMBModel.
2.4.1.6. EMBTerrainFinder
This class is responsible for providing a list of all available EMBTerrains in the EMBTerrainLibrary for the user to select.

```
public class EMBTerrainFinder {
    private EMBTerrain current;

    public EMBTerrainFinder(Object[] terrains) {
        // initialize with current terrain
    }

    public void searchDB() {
        // search for terrains
    }

    public void select() {
        // select terrain
    }

    public EMBTerrain getCurrent() {
        return current;
    }
}
```

2.4.1.7. EMBTerrainView
This class is responsible for providing a thumb-nail view of the currently selected EMBTerrain.

```
public class EMBTerrainView {
    private Canvas3D canvas3D;
    private int BOUNDSIZE;
    private Point3d USERPOSN;
    private SimpleUniverse su;
    private BranchGroup sceneBG;
    private BoundingSphere bounds;

    public void EMBTerrainView() {
        // initialize view
    }

    public void clearView() {
        // clear view
    }

    public void createSceneGraph() {
        // create scene graph
    }

    public void lightScene() {
        // light scene
    }

    public void addBackground() {
        // add background
    }

    public void initUserPosition() {
        // initialize user position
    }

    public void orbitControls() {
        // orbit controls
    }

    public void add(in s: BranchGroup) {
        // add to scene
    }

    public void setVisible(in v: boolean) {
        // set visibility
    }
}
```
2.4.1.8. EMBObjectPreview
This class is a container for the EMBObjectFinder and EMBObjectView. It will also add the currently selected EMBObject to the EMBBuildingSurface and EMBModel.

2.4.1.9. EMBObjectFinder
This class is responsible for providing a list of all available EMBObjects in the EMBObjectLibrary for the user to select.

2.4.1.10. EMBObjectView
This class is responsible for providing a thumb-nail view of the currently selected EMBObject.
2.4.1.11. EMBBuildingSurface

This class is responsible for displaying the top 2-D view of the current EMBEnvironment. From this view the user will be able arrange the objects and terrains that have been added to it. This view will also allow for removal of terrains and objects.

2.5. Model Package
2.5.1. Class Descriptions and Diagrams

2.5.1.1. EMBModel
This class is responsible for holding the current EMBEnvironment that is being built and making it available to other classes.

![Class Diagram for EMBModel]

2.5.1.2. EMBEnvironment
This class represents the current environment that is being built. It will be composed of numerous EMBTerrains and EMBObjects. It will also be responsible for building its XML definition.

![Class Diagram for EMBEnvironment]

2.5.1.3. EMBObject
This class is a collection of EMBBasicShapes that are to be used in the EMBEnvironment. It is also responsible for building its XML definition.
2.5.1.4. EMBBasicShape

This class is the super class for the primitive shapes; EMBBox, EMBCone, EMBSphere, and EMBCylinder. It is responsible for building the XML definition for the primitive shapes.
2.5.1.5. **EMBBox**

This class represents a box shape. It holds all the information necessary to represent a three dimensional box shape.

```
- length: double
- width: double
- height: double
- count: int
```

```java
EMBBox(x: double, y: double, z: double, l: double, w: double, h: double, f: float, g: float, b: float)
- getHeight(): double
- getWidth(): double
- getLength(): double
- writeXML(): String
```

2.5.1.6. **EMBCone**

This class represents a cone shape. It holds all the information necessary to represent a three dimensional cone.

```
- position: Point3d
- object_position: Point3d
- color: Color
- r: float
- g: float
- b: float
- h: double
- stationary: double
- direction: double
- name: String
```

```java
EMBBasicShape(in name: String): void
- getName(): String
- getR(): float
- getG(): float
- getB(): float
- getPos(): Point3d
- setHeight(in h: double): void
- setDirection(in d: double): void
- setStationary(in s: double): void
- getObjectPos(): Point3d
- setObjectPos(in p: Point3d): void
- move(in p: Point3d): void
- writeXML(): String
```
2.5.1.7. **EMBSphere**
This class represents a sphere shape. It holds all the information necessary to represent a three dimensional sphere.

```java
EMBSphere(radius: double, height: double, count: int)
```

- getRadius(): double
- getHeight(): double
- writeXML(): String

2.5.1.8. **EMBCylinder**
This class represents a cylinder shape. It holds all the information necessary to represent a three dimensional cylinder.

```java
EMBCylinder(radius: double, height: double, count: int)
```

- getRadius(): double
- getHeight(): double
- writeXML(): String

2.5.1.9. **EMBTerrain**
This class represents a terrain for the environment. It will consist of an elevation map and a collection of coordinates. The elevation map will specify the height of all the desired locations. The terrain will be represented by strips of triangles.
2.5.10. **EMBObjectLibrary**

This class will hold all the EMBObjects that are saved to the object library.

```java
public class EMBObjectLibrary {
    private objects: Hashable
    private libDir: File

    public EMBObjectLibrary(): void
    public query(name: String): EMBObject
    public getKeys(): Enumeration
    public addToLibrary(o: EMBObject, name: String): void
    public getData(): Enumeration
    public addToModel(o: String): void
    public parseXML(xmlFile: File): void
}
```

2.5.11. **EMBTerrainLibrary**

This class will hold all the EMBTerrains that are saved to the object library.

```java
public class EMBTerrainLibrary {
    private terrain: TriangleStripArray
    private shape: Shape3D
    private name: String
    private FLOOR_LEN: int
    private row[0..*]: int
    private green: Color3f
    private floorBG: BranchGroup
    private coords: ArrayList
    private ng: NormalGenerator
    private gi: GeometryInfo
    private points[0..*]: Point3f

    public EMBTerrain(name: String): void
    public createCoords(x: int, y: int, in coords: ArrayList): void
    public createGeometry(), coords: ArrayList, in col: Color3f): void
    public createAppearance(): void
    public updatePoint(x: int, y: Point3f): void
    public getBG(): BranchGroup
    public writeXML(): String
    public getHeight(index: int): float
    public getData(): Point3f
    public getName(): String
    public setName(name: String): void
}
```
2.6. Sequence Diagrams

The following sequence diagrams show some of the main functions that the Environment Model Builder will perform.

2.6.1. Opening a Saved Environment

The following sequence diagram show the sequence of actions involved in reading in a saved environment.

2.6.2. Adding a Terrain to the Building Surface

The following sequence diagram shows searching for a terrain and then adding it to the building surface.
2.6.3. Adding a Object to the Building Surface
The following sequence diagram shows searching for an object and then adding it to the building surface.

3. Environment Object Builder
The Environment Object Builder is a graphical tool for building complex shapes/object from primitive shapes. The tool will have three drawing surfaces representing a two dimensional view from the top, side, and front. The user will be able to move and resize the primitive shape from any of the three drawing surfaces. There will also be a three dimensional view provided to observe the created object in 3D. Finally there will be a XML view to show the textual description of object. The following sections will describe the packages of the Environment Object Builder

3.1. Package View
3.2. Application Package

3.2.1. Class Descriptions and Diagrams

3.2.1.1. EOBApplication
This class is just intended to have the main method for this program and create the EOBController and set it visible.
3.3. Controller Package

3.3.1. Class Descriptions and Diagrams

3.3.1.1. EOBController

This class is the main frame of the application. It will handle all the menu item actions. It is responsible for loading files, saving files to disk, and saving EOBObjects to the library.

```
EOBController

- view: EOBView
- model: EOBModel
- lib: EOBObjectLibrary
- desktop: JDesktopPane
- menuBar: JMenuBar
- fileMenu: JMenu
- newMenuItem: JMenuItem
- openMenuItem: JMenuItem
- exportXML: JMenuItem
- saveToLib: JMenuItem
- chooser: JFileChooser
- xmlFileName: String
- xmlFile: File

EOBController(): void
- addView(): void
- setSaveToLibMenuItemAction(): void
- setExportXMLMenuItemAction(): void
- setNewMenuItemAction(): void
- setOpenMenuItemAction(): void
- parseXML(): void
- exportXML(): void
- setZoomFactor(): void
```
3.3.1.2. EOBBoxPropertiesWindow
This class provides a JDialog window with controls for modifying a EOBBox.

```
EOBBoxPropertiesWindow
  length: JSpinner
  width: JSpinner
  height: JSpinner
  red: JSpinner
  green: JSpinner
  blue: JSpinner
  x: JSpinner
  y: JSpinner
  z: JSpinner
  weight: JSpinner
  box: EOBBox
```

EOBBoxPropertiesWindow(in s: EOBBasicShape)
  actionPerformed(in e: ActionEvent): void

3.3.1.3. EOBConePropertiesWindow
This class provides a JDialog window with controls for modifying a EOBCone.

```
EOBBoxPropertiesWindow
  ColorChangeListener
  ModifyChangeListener
  MoveChangeListener
  WeightChangeListener
```
### 3.3.1.4. EOBCylinderPropertiesWindow

This class provides a JDialog window with controls for modifying a EOBCylinder.
3.3.1.5. EOBFrontMouseHandler

This class is responsible for handling mouse events for the front building surface. In particular it will wait for mouse clicks and determine if one of the objects was clicked on. If an object is clicked on it will provide a properties window for that object.

3.3.1.6. EOBSideMouseHandler

This class is responsible for handling mouse events for the side building surface. In particular it will wait for mouse clicks and determine if one of the objects was clicked on. If an object is clicked on it will provide a properties window for that object.
3.3.1.7. EOBSpherePropertiesWindow
This class provides a JDialog window with controls for modifying a EOBSphere.

```
EOBSpherePropertiesWindow
- radius: JSpinner
- x: JSpinner
- y: JSpinner
- z: JSpinner
- red: JSpinner
- green: JSpinner
- blue: JSpinner
- weight: JSpinner
- sphere: EOBSphere
```

EOBSpherePropertiesWindow ninguém EOBBasicShape
- actionPerformed(in e: ActionEvent): void

3.3.1.8. EOBTopMouseHandler
This class is responsible for handling mouse events for the top building surface. In particular it will wait for mouse clicks and determine if one of the objects was clicked on. If an object is clicked on it will provide a properties window for that object.

```
EOBTopMouseHandler
- selected: EOBBasicShape
- view: EOBTOPDrawingView
```

EOBTopMouseHandler(in view: EOBTOPDrawingView):
mouseClicked(in e: MouseEvent): void
3.4. View Package

3.4.1. Class Descriptions and Diagrams

3.4.1.1. EOBView

This class is a container for the EOBThreeDimensionalView, EOBDrawingView, and EOBXMLView.

```
EOBView
- draw: EOBDrawingView
- xml: EOBXMLView
- threeD: EOBThreeDimensionalView
- pane: JTabbedPane
- panel: JPanel
```

3.4.1.2. EOBThreeDimensionalView

This class will show the three dimensional view of the current EOBOBJECT. From this view the user will be able to view the EOBOBJECT from any angle.
3.4.1.3. EOBDrawingView
This class is the container for the EOBTopDrawingView, EOBSideDrawingView, EOBFrontDrawingView, and EOBOBJECTPreview.

3.4.1.4. EOBXMLView
This class is responsible for displaying the XML definition of the current EOBOBJECT. The XML will represent the contents that will be saved to disk.
3.4.1.5. EOBSideDrawingView
This class is responsible for providing a drawing surface for EOBBasicShapes. This view will represent the side view. The user will be able to move and change the properties of the EOBBasicShapes from this view.

```
EOBSideDrawingView

EOBSideDrawingView(): void
paint(in g: Graphics): void
update(in t: Observable, in o: Object): void
update(): void
zoomIn(): void
zoomOut(): void
select(in p: Point2d): void
```

3.4.1.6. EOBFrontDrawingView
This class is responsible for providing a drawing surface for EOBBasicShapes. This view will represent the front view. The user will be able to move and change the properties of the EOBBasicShapes from this view.

```
EOBFrontDrawingView

EOBFrontDrawingView(): void
paint(in g: Graphics): void
update(in t: Observable, in o: Object): void
zoomIn(): void
zoomOut(): void
select(in p: Point2d): void
```

3.4.1.7. EOBTopDrawingView
This class is responsible for providing a drawing surface for EOBBasicShapes. This view will represent the top view. The user will be able to move and change the properties of the EOBBasicShapes from this view.

```
EOBTopDrawingView

EOBTopDrawingView(): void
paint(in g: Graphics): void
update(in t: Observable, in o: Object): void
zoomIn(): void
zoomOut(): void
select(in p: Point2d): void
```

3.4.1.8. EOBObjectPreview
This class is the container for the EOBObjectFinder and EOBObjectView. It will also add the currently selected EOBBasicShapes to the EOBSideDrawingView, EOBFrontDrawingView, and EOBTopDrawingView.
3.4.1.9. EOBOBJECTFinder
This class is responsible for providing a list of all available EOBOBJects in the EOBOBJECTLibrary for the user to select.

3.4.1.10. EOBOBJECTView
This class is responsible for providing a thumb-nail view of the currently selected EOBOBJECT.
3.5. Model Package

3.5.1. Class Descriptions and Diagrams

3.5.1.1. EOBModel

This class is responsible for holding the current EOBObject that is being built and making it available to other classes.
3.5.1.2. EOBObject
This class is a collection of EOBBasicShapes that are to be used in the environment. It is also responsible for building its XML definition.

3.5.1.3. EOBBasicShape
This class is the super class for the primitive shapes; EOBox, EOBCone, EOBSphere, and EOBCylinder. It is responsible for building the XML definition for the primitive shapes.
3.5.1.4. EOBBox

This class represents a box shape. It holds all the information necessary to represent a three-dimensional box shape.
3.5.1.5. EOBCone
This class represents a cone shape. It holds all the information necessary to represent a three dimensional cone.
3.5.1.6. EOBSphere

This class represents a sphere shape. It holds all the information necessary to represent a three dimensional sphere.
3.5.1.7. EOBCylinder
This class represents a cylinder shape. It holds all the information necessary to represent a three dimensional cylinder.

```
EOBCylinder

- top: Double
- side: Double
- front: Double
- height: double
- local_height: double
- radius: double
- local_radius: double
- count: int
```

```
EOBCylinder(in x: double, in y: double, in z: double, in r: double, in h: double, in red: float, in g: float, in b: float,
  drawFront(in g: Graphics2D): void
  drawSide(in g: Graphics2D): void
  drawTop(in g: Graphics2D): void
  get Höhe(): double
  getRadius(): double
  rebuild(): void
  modify(in h: double, in r: double): void
  writeXML(): String
  getTopBounds(): Rectangle2D
  getSideBounds(): Rectangle2D
  getFrontBounds(): Rectangle2D
```

3.5.1.8. EOBObjectLibrary
This class will hold all the EOBObjects that are saved to disk.

```
EOBObjectLibrary

- objects: Hashtable
- libDir: File
```

```
EOBObjectLibrary(): void
addToModel(in obj: String): void
query(in name: String): EOBObject
getKeys(): Enumeration
addToLibrary(in o: EOBObject, in name: String): void
exportXML(in string: String): void
getData(): Enumeration
parseXML(in xmlFile: File): void
```

3.6. Sequence Diagrams
The following sequence diagrams show some of the main functions that the Environment Object Builder will perform.
3.6.1. **Modifying and Moving A Box Shape**

The following sequence diagram show selecting a box and moving it. It also shows selecting a box and modifying its dimensions.

4. **Environment Terrain Builder**

The Environment Terrain Builder is a graphical tool for building surfaces to be used by the Environment Model Builder. The tool will provide a building surface to allow the user to specify the elevation of a given region on the surface. The user will also be able to define physical properties of the surface. A three dimensional view will be provided to observe how the terrain will look in 3D. Finally a XML view will be provided to give a textual description of the terrain. The following sections will describe the packages of the Environment Terrain Builder.

4.1. **Package View**
4.2. Application Package

4.2.1. Class Descriptions and Diagrams

4.2.1.1. ETBAplication
This class is just intended to have the main method for this program and create the
ETBController and set it visible.

4.3. Controller Package

4.3.1. Class Descriptions and Diagrams

4.3.1.1. ETBController
This class is the main frame of the application. It will handle all the menu item actions. It is
responsible for loading files, saving files to disk, and saving ETBTerrains to the library.
4.4. View Package

4.4.1. Class Descriptions and Diagrams

4.4.1.1. ETBView
This class is a container for the ETBThreeDimensionalView, ETBDrawingView, and ETBXMLView.

4.4.1.2. ETBThreeDimensionalView
This class will show the three dimensional view of the current ETBTerrain. From this view the user will be able to view the ETBTerrain from any angle.
4.4.1.3. ETBXMLView
This class is responsible for displaying the XML definition of the current ETBTerrain. The XML will represent the contents that will be saved to disk.

```
ETBXMLView
```

4.4.1.4. ETBDrawingView
This class is a container for the ETBBuildingSurface and ETBTerrainPreview.

```
ETBDrawingView
```

4.4.1.5. ETBBuildingSurface
This class is responsible for providing a surface to create an ETBTerrain. It will provide the user an interface to specify the elevation of sections of the ETBTerrain.
4.4.1.6. ETBTerrainPreview
This class is a container for the ETBTerrainView and ETBTerrainFinder. It will provide the ability to add the currently selected ETBTerrain to the ETBBuildingSurface.

4.4.1.7. ETBTerrainFinder
This class is responsible for providing a list of all available ETBTerrains in the ETBTerrainLibrary for the user to select.

4.4.1.8. ETBTerrainView
This class is responsible for providing a thumb-nail view of the currently selected ETBTerrain.
4.5. Model Package

4.5.1. Class Descriptions and Diagrams

4.5.1.1. ETBModel
This class is responsible for holding the current ETBTerrain that is being built and making it available to other classes.
4.5.1.2. ETBTerrain

This class represents a terrain for the environment. It will consist of an elevation map and a collection of coordinates. The elevation map will specify the height of all the desired locations. The terrain will be represented by strips of triangles.

4.5.1.3. ETBTerrainLibrary

This class will hold all the EOBTerrains that are saved to disk.
4.6. Sequence Diagrams
The following sequence diagram shows the main function of the Environment Terrain Builder.

4.6.1. Modifying the Elevation of a Section of the Terrain
The following sequence diagram shows how a user would set the elevation for a section of the terrain.

5. Formal Specification for the Environment Model Builder
The formal specification is limited to the Environment Model Builder.

5.1. USE Model

model emb

--
-- CLASSES
--
class Point2d
end
class Point3d
end
class File
end
class Shape3D
end

--
-- APPLICATION PACKAGE
--
class EMBApplication
operations
  inti()
end

--
-- CONTROLLER PACKAGE
--
class EMBController
operations
  save(f : File)
  open(f : File)
  exportXML()
  parseXML()
  setZoomFactor()
end

--
-- VIEW PACKAGE
--
class EMBView
end
class EMBThreeDimensionalView
operations
  add(s : Shape3D)
end
class EMBXMLView
operations
  showXML()
end
class EMBDrawingView
end
class EMBBuildingSurface
  operations
  paint()
  zoomOut()
  zoomIn()
  select(p : Point2d)
end

class EMBOBJECTPreview
  operations
  addObject()
end

class EMBOBJECTView
  operations
  display(s : Shape3D)
end

class EMBOBJECTFinder
  operations
  searchDB()
  getCurrent() : EMBOBJECT
  select()
end

class EMBTerrainPreview
  operations
  addTerrain()
end

class EMBTerrainView
  operations
  display(s : Shape3D)
end

class EMBTerrainFinder
  operations
  searchDB()
  getCurrent() : EMBTerrain
  select()
end

--
-- MODEL PACKAGE
--

class EMBEnvironmentLibrary
  operations
  getData() : Set(EMBEnvironment)
  addData(e : EMBEnvironment)
end

class EMBOBJECTLibrary
  operations
  getData() : Set(EMBOBJECT)
class EMBObject
attributes
   name : String
   x : Real
   y : Real
   z : Real
   length : Real
   width : Real
   height : Real
operations
   writeXML() : String
   addShape(s : EMBBasicShape)
   move(p : Point3d)
end

class EMBBasicShape
attributes
   name : String
   x : Real
end

class EMBTerrain
attributes
   name : String
operations
   getHeight(x : Real, z : Real) : Real
   writeXML() : String
end

class EMBModel
operations
   getData() : EMBEnvironment
   addObject(o : EMBObject)
   addTerrain(t : EMBTerrain)
   deleteObject(o : EMBObject)
   deleteTerrain(t : EMBTerrain)
end

class EMBEnvironment
attributes
   name : String
operations
   writeXML() : String
   addObject(o : EMBObject)
   addTerrain(t : EMBTerrain)
   deleteObject(o : EMBObject)
   deleteTerrain(t : EMBTerrain)
end

class EMBTerrainLibrary
operations
   getData() : Set(EMBTerrain)
   addData(t : EMBTerrain)
end

addData(o : EMBObject)
end
y : Real
z : Real
operations
writeXML() : String
end

class EMBBox < EMBBasicShape
attributes
  length : Real
  width : Real
  height : Real
end

class EMBCylinder < EMBBasicShape
attributes
  height : Real
  radius : Real
end

class EMBCone < EMBBasicShape
attributes
  height : Real
  radius : Real
end

class EMBSphere < EMBBasicShape
attributes
  radius : Real
end

--
-- ASSOCIATIONS
--

-- VIEW PACKAGE
--

association ThreeD between
  EMBView[1]
  EMBThreeDimensionalView[1] role threeD
end

association Canvas between
  EMBView[1]
  EMBDrawingView[1] role canvas
end

association XML between
  EMBView[1]
  EMBXMLView[1] role xml
end

association TerrPreview between
EMBDrawingView[1]
EMBTerrainPreview[1] role terrainPrev
end

association Builder between
EMBDrawingView[1]
EMBBuildingSurface[1] role builder
end

association ObjPreview between
EMBDrawingView[1]
EMBOBJECTPREVIEW[1] role objectPreview
end

association TerrainList between
EMBTerrainPreview[1]
EMBTerrainFinder[1] role finder
end

association TerrainThumbNail between
EMBTerrainPreview[1]
EMBTerrainView[1] role view
end

association ObjectList between
EMBOBJECTPREVIEW[1]
EMBOBJECTFINDER[1] role finder
end

association ObjectThumbNail between
EMBOBJECTPREVIEW[1]
EMBOBJECTVIEW[1] role view
end

--
-- MODEL PACKAGE
--

association Model between
EMBMODEL[1]
EMBENVIRONMENT[1] role environment
end

association EnvDatabase between
EMBENVIRONMENT[1..*] role environments
EMBENVIRONMENTLIBRARY[1]
end

association Surface between
EMBENVIRONMENT[1]
EMBTERRAIN[1] role terrain
end

association Objects between
EMBENVIRONMENT[1]
EMBOBJECT[1..*] role objects ordered
association TerrainDatabase between
  EMBTerrain[1..*] role terrains
  EMBTerrainLibrary[1]
end

association ObjectDatabase between
  EMBObject[1..*] role objects
  EMBObjectLibrary[1]
end

association Shapes between
  EMBObject[1]
  EMBBasicShape[1..*] role shapes
end

--
-- CONSTRUCTS
--

constraints
--
-- Relations
--

--
-- Unique names of environments in Environment Library
--
context e : EMBEnvironmentLibrary
inv UniqueNameEnvironmentLibrary:
  e.environments->forAll(p1,p2 | p1 <> p2 implies p1.name <> p2.name)

--
-- Unique names of objects in Object Library
--
context o : EMBObjectLibrary
inv UniqueNameObjectLibrary:
  o.objects->forAll(p1,p2 | p1 <> p2 implies p1.name <> p2.name)

--
-- Unique names of terrains in Terrain Library
--
context t : EMBTerrainLibrary
inv UniqueNameTerrainLibrary:
  t.terrains->forAll(p1,p2 | p1 <> p2 implies p1.name <> p2.name)

--
-- Unique names for all shapes of an object
--
context obj : EMBObject
inv UniqueNameObjectShapes:
  obj.shapes->forAll(p1,p2 | p1 <> p2
implies p1.name <> p2.name)

--
--Every box has positive length, width and height
--
context b : EMBBox
  inv BoxPositiveLength:
    b.length > 0
  inv BoxPositiveWidth:
    b.width > 0
  inv BOXPositiveHeight:
    b.height > 0

--
--Every sphere has positive radius
--
context s : EMBSphere
  inv SpherePositiveRadius:
    s.radius > 0

--
--Every cylinder has positive height and radius
--
context cyl : EMBCylinder
  inv CylinderPositiveHeight:
    cyl.height > 0
  inv CylinderPositiveRadius:
    cyl.radius > 0

--
--Every cone has positive height and radius
--
context c : EMBCone
  inv ConePositiveHeight:
    c.height > 0
  inv ConePositiveRadius:
    c.radius > 0

--
-- Operations
--

-- Deleting an object must remove it while the other object are unchanged
context EMBEnvironment::deleteObject(o : EMBObject)
  pre  Current: objects->includes(o)
  post Deleted: objects = objects@pre->excluding(o)

-- Added objects must be unique
context EMBEnvironment::addObject(o : EMBObject)
  pre  Current: objects->excludes(o)
  post Added:  objects = objects@pre->including(o)
--Additional OCL statements at request of Committee

--Elevation adjustment
Context EMBEnvironment::elevationAdj(o:EMBObject)
   Post adjusted: o.y = terrain->getHeight(o.x,o.z)

--Shapes with in Object bounds
context o : EMBObject
   inv bounds:
      shapes->forall( s | (o.x – o.length/2 < s.x < o.x + o.length/2) and (o.y –
                     o.height/2 < s.y < o.y + o.height/2) and (o.z – o.width/2 < s.z < o.z +
                     o.width/2))

--Move to back
context EMBEnvironment::moveToBack(o:EMBObject)
   post back : objects->last = o
   post size : objects@pre->asSet() = object->asSet()