CSS450 – Computer Graphics

UFOh No! Level Editor

Ryan Hoaglan, Cameron Rahman, Sid Maxwell

Table of Contents

Introduction

The UFOh No Level Editor will be designed to allow users to create their own levels to play in with the UFOh No video game. Users will be able to create, modify, and save maps in xml format that can be loaded by the video game application. The main controls of the editor will be a toolbox of available objects that the user will be able to drag and drop into the world to define the map. Individual objects' properties will be allowed to be modified once they have been placed within the level.

Functional Specifications

Hierarchy and Transformation

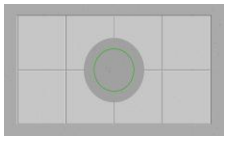
Each object that the user will be able to interact with will be comprised of scene nodes that would allow translation transformations. Child scene nodes of the overall object will all move with the parent when the user changes its location about the level. In some cases, the scene nodes will need to be rotated with the root node for objects that have a directional property.

When the user chooses to save the map they are working on, the transforms will be applied to the objects in the world to determine their coordinates. These positions and orientations will be output into xml format that would be able to be read from the UFOh No game.

Primitive Attributes

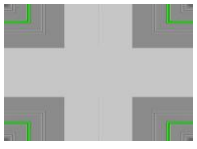
Every object that the user can place within the level will have a texture mapped to reflect a similar appearance to what that object represents in the UFOh No game. The level editor will also have an option of animating the aliens in which the texture maps would be updated constantly. In addition, each room's environment will be texture mapped to look like the rooms in the game.

Room Textures

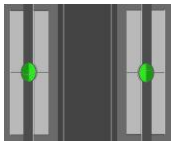


Background

Wall Segment



Corner Piece



Door

Goal

Start

Character Textures


Floating Alien

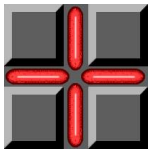

Vertical/Horizontal Alien


Sleeping Alien

Room Objects


Key


Movable Block


Move Once Block


Stationary Block

Complexity

The majority of transforms required by this application will be the translation of various objects around each room. It is for this reason that scene nodes will not be used to control the placement and orientation of the objects. When the user rotates an element (if applicable) the change will be reflected in the change of a directional arrow hovering over that object.

Predefined Animation

Animate objects will be shown similar to how they appear in the UFOh No game. Once placed into a room, if the user has selected the option, aliens will move about how they would if the user was playing the game. When moving, their texture maps will be changed according to their associated sprite sheets.

Camera Control and Multiple Views

The main world view will display the room within the level the user is currently editing. In this view, users will be able to place various objects around the room. An additional, smaller world view will be used to show all the rooms of the entire level. Clicking on this view will change the focus of the main view to the chosen room within the level.

Semantic of Application

The purpose of this application is to create xml files that can be loaded into the UFOh No video game to map out the levels the player will interact with. The level editor will offer an intuitive user interface to the manipulation of this data file. Any user will be able to load, modify, and export levels in the form of a valid xml file for the game to read from upon initialization.

Suspend/Resume/Reset Support

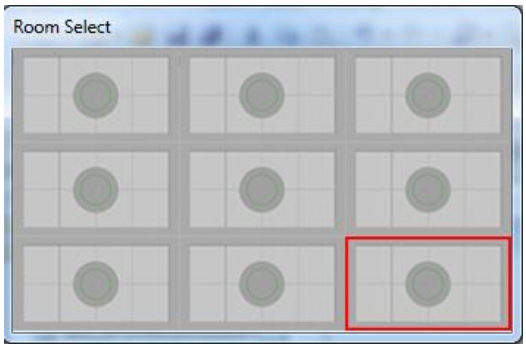
The application’s current state can be loaded, modified, and saved. A new state can also be generated if desired. When the application loads a state, the xml file will be chosen by the user to read and will be parsed into objects

to populate the level. When the user chooses to save the current state of the model, an xml file will be created (with a specified filename) containing the current state's data.

GUI Layout

Room Selector GUI

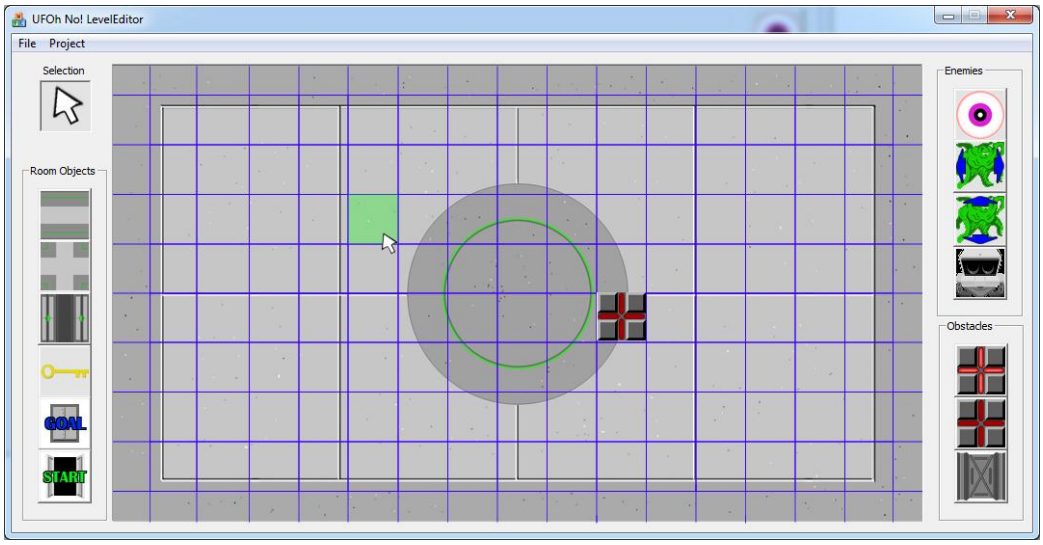
The Room Selector GUI will be a floating dialog window that will always remain on top of the main UFOh No! Level Editor window. The user will be able to move around the Room Selector GUI, positioning it wherever is the most convenient. Within this window, the user will be able to change the current editing room.



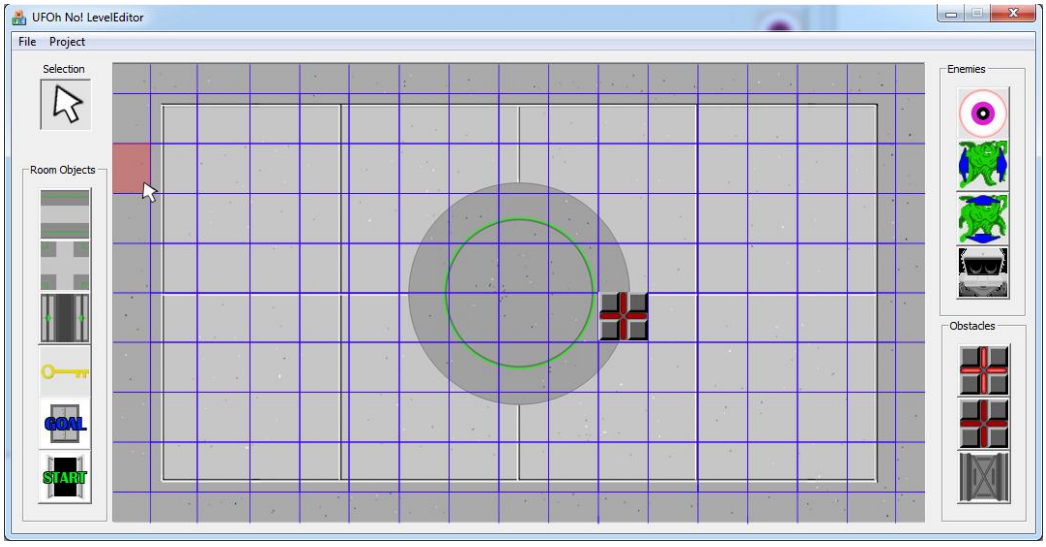
UFOh No! Level Editor GUI

All editing takes place in the main UFOh No! Level Editor GUI. There are three sets of object toolbars, each containing a different type of object that the user can add to the level: Enemies, Obstacles, and Room Objects. Enemies are objects that can interact with the user, causing a loss of life if contact is made. Obstacles are blocks that the user can interact with. Finally, Room Objects are objects such as walls, doors, and start/stop positions that make up the room. There is an additional selector icon that allows the user to select and move a currently placed object to a different grid square.

Once the user clicks on an object icon, it will deselect the previously selected icon, make the new object active, and depress the icon of the current object to better identify which object has been selected. In the following picture, the user is moving their mouse cursor inside the D3D editing window. The green box alerts the user that the square is capable of supporting the currently selected object. Once the user clicks their left mouse button on a valid square, the currently selected object is placed into that square. The user now has the option of placing an additional object, or changing the currently selected object to something else.



Not all grid positions support each object type. When the user hovers their mouse cursor over an invalid square, the green box changes color to red. This indicates to the user that they are unable to place an object of the currently selected object type inside of the square. The following picture indicates that the user is trying to place an object in an invalid grid square. The user's left click will do nothing if the square is invalid



Object Interaction

When placing the objects about the level, the destination location will be decided based on the nearest available unit of area to the cursor. The new object's bounding box will need to interact with those of the objects currently

in the room already to ensure that, in most cases, the object is not placed on top of an existing item. However, some objects are able to be placed on the same grid square (for example a key may be placed under a moveable block) so in some cases overlapping bounding boxes will be allowed.

When animation mode is selected, the objects in the room will behave as if they were in the game. This will require collision detection between moving aliens, blocks, and walls. Their collision reaction will be identical to how they would respond during gameplay.

View/Controller Pair Support

The main world view will display the room within the level the user is currently editing. In this view, users will be able to place various objects around the room. Mouse hovering over this view will show the currently selected object from the toolbar directly underneath the mouse cursor. An outlined box will be shown in the room depicting where, if clicked, the object will be placed. This placement box will snap to the room grid units to ensure no overlapping is allowed. When the user does not have an object from the toolbar selected, clicking on the main view will grab whatever object the cursor is currently over and allow the user to drag it to a new location within the room.

The smaller world view will be used to show all the rooms of the entire level. Clicking on this view will change the focus of the main view to the chosen room within the level. When the room is changed, the main view's world window will snap to the room dimensions to only display one room at a time.