Goblin City
An AR 3D City Modeling Tool

Amrita Mazumdar am3210@columbia.edu Kathy Sun xs2161@columbia.edu
Olivia Winn oaw2102@columbia.edu
COMS 4172 - 3D User Interfaces & Augmented Reality

Professor: Steven Feiner

May 10, 2013

Contents

1 Introduction

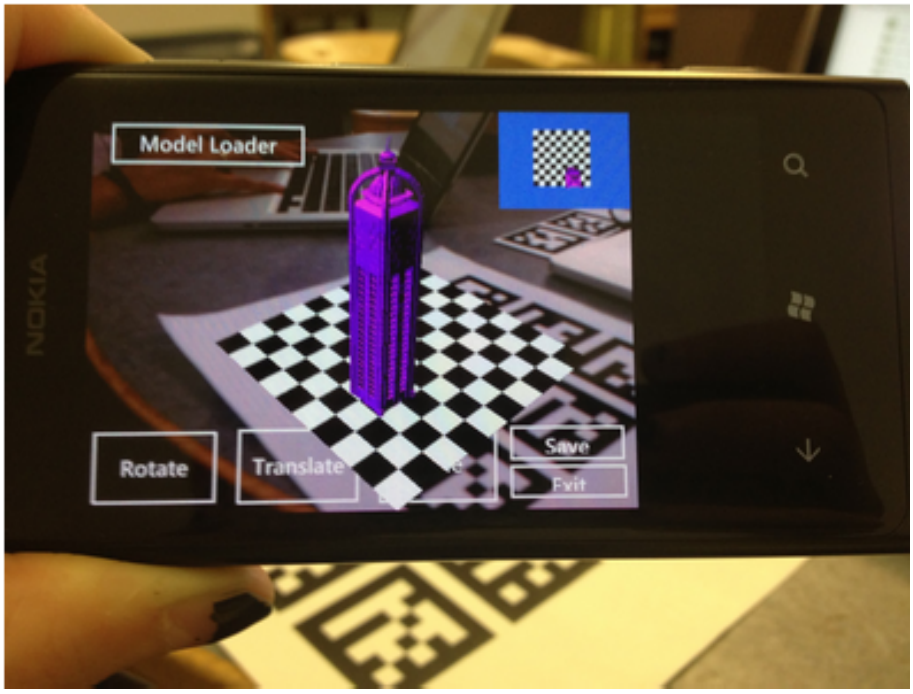

Figure 1: A skyscraper in GoblinCity.

GoblinCity is a augmented reality tool designed to help users build 3D city models. It is a Windows 7 phone application designed to allow users to construct cities from pre-made models using interactions in physical space rather than a virtual 2D screen. This approach reduces the complexity of the user interface needed to support building selection, translation, rotation, and scaling to just a couple of buttons for function selection and a simple physical controller.

2 System Overview

GoblinCity opens with a menu allowing the user to choose whether they would like to start a new city, load an old one, or check out the info page for tips and reminders on how to use the program. Choosing a load screen (building model or city) takes the user to a new page where they can scroll through images of their potential choices. Once a building model is chosen, the user can then import it into their scene and manipulate it in Creative mode. In Creative mode, the user is presented with an AR view of their current city, a top-down minimap view, and a set of buttons for building manipulation. The user can select any building in their view and transform the building via scaling, translation, and rotation. A base grid is drawn to indicate the "ground" for the user, and the size of the city is only conditional upon the size of the ground array used.

3 3D Interaction Feature Evaluation

3.1 Selection & Manipulation

Users manipulate buildings using ARtag fiducial markers for building viewing and manipulation. A ground plane optical marker array establishes a common 3D coordinate system for the buildings to be placed and manipulated upon, and an optical marker toolbar is used for manipulating a selected building in the user's physical space. Selection is handled by the Matali Physics engine in the phone. Once a building is selected, a contextual set of buttons to allow for building transformation appears. The user can only manipulate a building if it has been selected and an explicit building transform has been indicated. Only one type of building transformation can occur at a time.

3.1.1 Selection

Users select buildings by tapping the building image on the phone screen. The user identifies a selected building through ray casting. Only one building can be selected at any given time. This pointing-based isomorphic approach was chosen because it is accurate and integrates smoothly in the user action flow. Users physically navigate themselves to a position where they have a good view of the building and select the building on their phone screen. The user receives feedback during their selection task from a color change in the selected building. The building changes colors from `purple`, the default building color, to `green`, the chosen selection color, to indicate that it has been selected. The user can then tap the button associated with the desired manipulation mode to active a manipulation technique, and the button will turn `red` to indicate selection (in contrast with the usual `clear` appearance). The building also changes color in the top-down minimap view, to further reinforce confirmation of a selected building. The advantages of this technique are simple implementation, ease and efficiency of use for the user, and effective and accurate selection.

We also considered using toolbar intersection for building selection. This would have worked

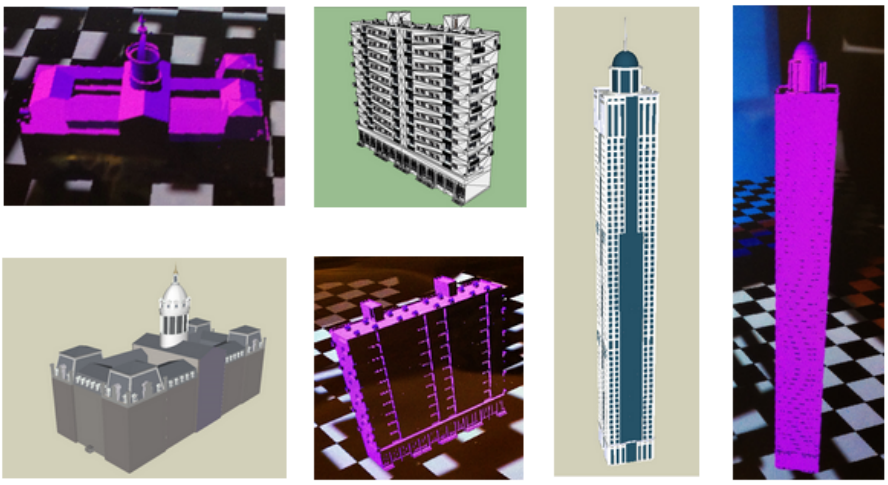

Figure 2: Various buildings provided in GoblinCity.

by using the camera to track an AR toolbar, and selecting the building that the position of the

AR toolbar intersected with. This approach has the advantage of being more physical which is in line with GoblinCity's original goal of moving building manipulation into physical space. However, in practice, this approach was not sufficiently accurate because the relatively large toolbar would frequently select an incorrect building. Additionally, tapping through the screen intuitively integrates into the user flow because the user must view the screen to verify the desired building is selected.

3.1.2 Translation

Once a building is selected, it is possible to move the selected building in the any direction in the `x-y` coordinate plane using an ARtag toolbar. To translate a building, the user selects a building, then selects the "Translate" button. The building will jump to the location of the ARtag toolbar and move where the toolbar moves. The user then moves the ARtag toolbar within the view of the phone camera and the ground array to translate the building around on the city grid.

Users are not allowed to move the selected building up and down in the `z` (vertical) direction

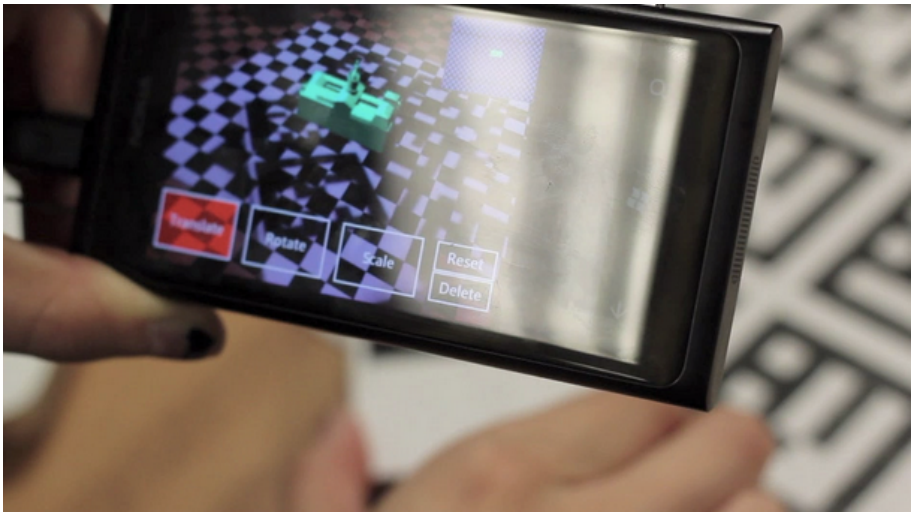

Figure 3: A user translating a skyscraper.

because this would violate real-world city-building constraints (floating buildings, while visually interesting, are not realistic). All translations are discrete. The buildings snap to grid which ensures accurate building placement and alignment. To set a building in place, the user taps outside of the building on the screen, either in empty space, on another building, or another transform button, and the building's position will freeze.

3.1.3 Rotation

Buildings can be rotated in any direction, using the ARtag toolbar to guide the rotation of the building. To rotate a building, the user selects a building, then selects the "Rotate" button. The user then moves the ARtag toolbar within the view of the phone camera and the ground array and rotates the toolbar to their liking. The building will simultaneously rotate with the toolbar.

Another technique considered for rotation was handling a building's rotation through the rotation of the user's camera. This eventually proved to be confusing for the user, interfering with their ability to view the ARtag ground array, and went against the natural physical manipulation the program is emulating.

3.1.4 Scaling

Buildings can be uniformly stretched or compressed using the ARtag toolbar. To stretch a building, the user selects a building, selects "Scale", and moves the ARtag toolbar closer or further away from the selected building. Moving the ARtag toolbar away from the building makes the building larger, as if the user were physically stretching the building to enlarge it. Moving the toolbar closer to the building shrinks it as if the user were physically compressing the building.
We also considered scaling objects by the position of the camera, with the user moving forward or backwards with respect to the ground array and city scene to delineate the scale factor. Although this approach worked flawlessly, it appeared to be at odds with the travel and wayfinding capabilities implemented for user navigation, so we eliminated this method and chose the above approach.

3.2 Travel

Travel and movement throughout the GoblinCity environment is extremely intuitive and flexible, with the user physically free to move their camera to view any part of the city on the ground array and walk around without much hindrance. In our implementation, travel is a primary task for the user, as building manipulation is the real main task and travel is almost required for the precise and complex building arrangements users expect from an AR city layout tool. The user takes an active role in movement, with the only restrictions being that of the size of their ground array.

3.3 Wayfinding

Wayfinding is implemented artificially through a small mini-map in the upper-right corner of the creative mode view. The buildings in the current city can be seen in a top-down orientation, with a small red pyramid indicating the position and orientation of the user. As the user manipulates buildings in their 3D world space, the building moves in the minimap view as well, helping users orient and maneuver themselves through the buildings to best suit their creative needs. If the user moves the phone outside of the view of the minimap, the pyramid representing the user stays at the edge of the bounds of the minimap, so the user can still get a sense of their position and orientation.

When considering other wayfinding techniques, a minimap was the most obvious and appropriate solution for wayfinding within the GoblinCity space. In the Creative Mode, there are a significant number of control buttons, so screen space is at a premium. For this reason, we could not implement a more complex wayfinding implementation such as reference objects, AR compasses, signage and trails, or audio cues without obstructing critical parts of the GoblinCity scene itself.

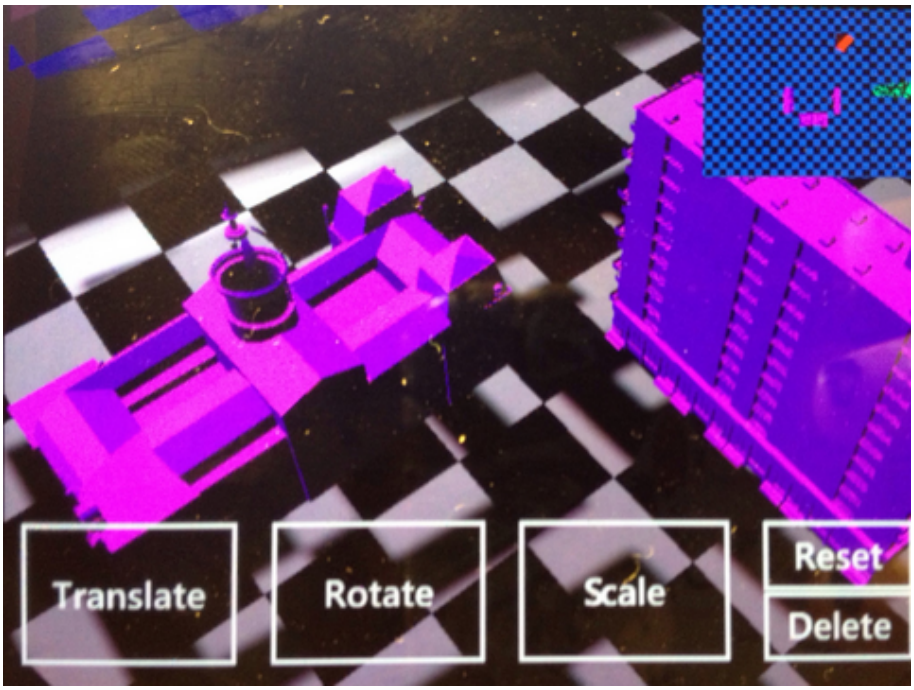

Figure 4: A close-up view of a populated city and minimap.

3.4 System Control

GoblinCity has several different views that lend itself for efficient city creation. The program opens up in the System Navigation mode, where users can view their different building options and load them in. The user can then navigate into Creative mode, where the city's buildings can be moved and the user can explore the city. In each mode, we have a few different contextual menus that help the user

3.4.1 Creative Mode Menus

The Creative Mode Menus exist purely within the space of the Creative Mode view. The user is initially presented with a set of navigational buttons to other modes in the program, such as the Model Loader. Upon selection of a building, the menus change to reflect the updated system status and the user is then free to transform the selected building. This contextually loaded menus provide freedom in user control as well as error prevention, and preserve precious screen real estate for city creation.

3.4.2 System Navigation Menus

The System Navigation menus allow the user to go between different modes of use. From the main menu, the user can go to a loading page to choose either a building model or a city to begin working on. If the user chooses `Back` from either page, they are taken back to the Main Menu. However, if the user had made a choice and progressed to the Edit View, the `Back` button of the loading pages would then redirect the user back to the Editing View, rather than the Main Menu.

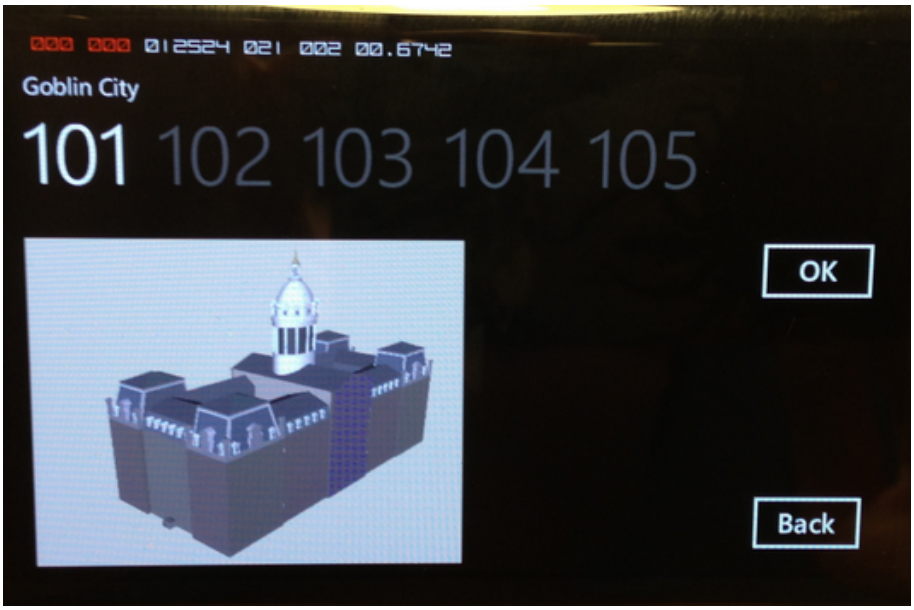

Figure 5: The Model Loader menu set.

3.5 Display & Device Integration

GoblinCity is implemented using GoblinXNA, a 3D software framework based on Microsoft XNA Game Studio. GoblinXNA supports external networking, vision based tracking, and physical simulation libraries. Ray casting is implemented through Matali Physics. The system status backend was implemented using Python, the Flask framework, and Heroku. Software development was done in a combination of Vim and Visual Studio 2012.

4 User Interface Heuristics Evaluation

4.1 System Status Visibility

GoblinCity updates visually updates users in real time using color changes to indicate building state changes and a dynamic menu system to indicate user mode changes. Feedback about building manipulations is provided on the phone immediately after the system processed the user action. Selected buildings change color. User actions that translate, scale, and rotate buildings are immediately reflected in the position and size of the selected buildings. Feedback about the mode the user is in is reflected in the visible menu. When a user has selected a building, the user is put into manipulation mode where the menu options change from system level actions such as save, exit and undo to building level actions such as translate, rotate, and scale. After a user choses a manipulation mode, that mode is highlighted in red on the menu. System data can be viewed online so more advanced users can see internal system status as well.

4.2 System-Real World Correlation

GoblinCity's primary user actions are designed to not only match but mirror the real world. Application navigation uses buttons that are labelled with real world terms like next, back, and

new. Object manipulation is controlled with a physical real world toolbar. As users move the toolbar, the objects in the application mirror the toolbar's movements.

4.3 User Control & Freedom

Users are free to move their buildings wherever they desire within the view of the camera and the grid space. To assist users when they unexpectedly enter an undesired menu or control panel, `exit` and `back` buttons are provided in each view. Although we explored undo-redo functionality for GoblinCity, it could not be effectively implemented at the time of this writing.

4.4 Consistency & Standards

GoblinCity is designed to be internally consistent with its program conventions as well as externally consistent with mobile application and Windows Phone 7 conventions. Within GoblinCity, users change modes by selecting buttons and control object manipulations using a ARtoolbar. Each action is associated with one button and each button is labelled with the affiliated action. In general, GoblinCity follows mobile application best practices by using large buttons and standard mobile design conventions such as placing titles at the top and menus at the bottom. Additionally, though the buttons available to the user changes depending on whether a building has been selected, the appearance of the buttons is exactly the same in either state, so the user does not have to search for the buttons or reorient themselves in any way.

4.5 Error Prevention

GoblinCity's approach to error prevention to restricting user interaction is to the set of moves that the system has verified as legal. At each step, user's actions are restricted to the buttons they can press and the system only generates buttons that allow legal actions. Within each action, the user provides information to the system through an ARmarker whose position is used to generate values that the system uses to update the city model. Because the system generates the values, the user will not be able to provide the system invalid input.

4.6 Recognition vs Recall

GoblinCity helps users navigate the application by using a dynamic menu system which shows the user all of the available options and limits the menus presented to functions that relate to the state of the system (i.e., when a building is selected, building transformations are available, and when the user is simply viewing the city they can choose to navigate to the model loader to add more buildings). Because all of the relevant options are shown at each step, users do not need to learn to navigate any menu systems or remember the options they have. As mentioned earlier, this also helps in error prevention and minimizes screen space used for button-based controls.

4.7 Flexibility & Efficiency of Use

In addition to the GUI provided for manipulation, we give users access to the raw city data online. All city data including recent moves and city configurations are uploaded to a server online. More advanced users can numerically and programmatically make tweaks to their cities.

This allows advanced user to rapidly make a large number of similar cities as well as have total control over the exact position of buildings.

4.8 Aesthetic & Minimalist Design

GoblinCity is designed to guide the user through each decision and provide information with overwhelming the user with options. The user interactions are split into multiple pages where the user is asked to make simple decisions, often just a single decision, at each point and all the relevant information is presented on each page. The user starts on the landing page where he or she can make a new city or load an existing city. In city loader page, the user can chose a city to load. In the new city page, the user can select a model to place in an empty city. A city (either preloaded or with one model) is selected, the user enters creative mode where he or she can see the city and select a model to manipulate. Once a model is selected for manipulation, the user can select the manipulation he or she wants to perform and execute that manipulation.

4.9 Error Recognition, Diagnosis, & Recovery

While the system is designed to prevent errors, errors associated with GoblinXNA frameworks, such as NyAR Toolbar Kit and Matali Physics, are out of GoblinCity's control. If one of these error occurs, GoblinCity will autosave the most recent state and crash. When the user returns to GoblinCity, he or she can load up the most recent city and seamlessly continue his or her work.

5 Acknowledgements

The GoblinCity team would like to thank Professor Feiner for a great semester and the opportunity to work with the GoblinXNA framework, and Carmine Elvezio and Mengu Sukan for their infinite GoblinXNA wisdom and advising.