Chapter 4 **Combining media into the system**

In this step, this research implements the combination of the media. Considering sketches, manually made models, and digital models are three of the most commonly used design media. Also, lots of the developments of new design media are based on these three media. This research chooses these three to be combined into the system in this step. For physical media, sketches and models, the research chooses the devices for capturing data from them and then implements the functions to decode, analyze, synchronize, and transform the data. In other words, physical models and sketches are combined into the system. On the other hand, digital models are combined without using hardware devices. The direct manipulation for modifying digital model is not included in this step. However, some elements for evaluating the design such as shadows and traffic conditions are added on the digital models. The manipulation of these elements is also added into the system in this step.

As mentioned in the previous chapter, the user interfaces for designers to adjust the settings about these three media are also implemented by extending the GUI prototype made in the first step. Figure 4.1 illustrates what are needed for combining these three media, which are implemented in this step. In figure 4.1, module (1) shows the three media that are combined in this step. Module (2) is the hardware devices or the software to capture the data from the media. Module (3) shows the data that are extracted from the capturing data. Some of these data are for displaying and some are for synchronizing. Module (4) is the rendering process of each medium and it would be connected to one or more canvases (module (5)).

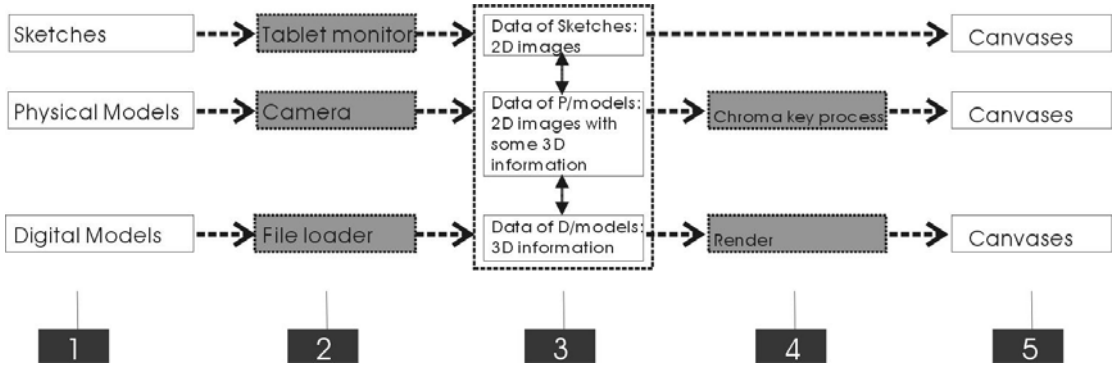

Figure 4.1: (1) media data (2) media data receive process (3) media data analyze and synchronize (4) media data transform (5) data storage (canvases)

The method this research uses for combining these three media only captures parts of the media data. Take the integration of physical models for example. It would be better if the system can capture the full 3D data of the models. But due to the technology for real-time 3D scanning is not matured; only the 2D images of the models can be captured in this research. However, since the capturing of each medium data is independent from the system and other media, these parts could be replaced or improved in the future when the technology for capturing that data is matured.

4.1 Combining models

The author first makes describes the integrating of digital and physical models. This includes the importing, decoding, and synchronizing of data from both physical and digital models.

4.1.1 Capturing physical model data

For the data of 3D physical models, this system uses a web camera to capture the 2D image of the physical models. Two different data are extracted from these captured images. One is the image of the physical models; the other is the coordinates of the physical camera. As we can see in figure 4.1.1, two threads process simultaneously after the image is grabbed. The image captured by the camera needs to be applied with image processing to extract the images of the physical models. And the coordinates of the physical camera is for synchronizing with the camera of the digital models.

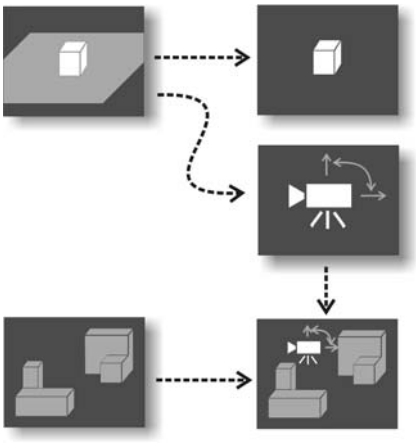

Figure 4.1.1: Two data are extracted from the image grabbed by the camera. One of them is for synchronizing with digital models.

4.1.2 Image processing of physical model image

As shown in figure 4.1.1, the image processing thread is being processed after the image is captured. The purpose of this thread is to remain the image of the physical models and eliminate the background. Background has to be eliminated to make the physical models possible to integrate into the digital models. The system uses Chroma Key processing to do the background elimination. To improve the accuracy of the background elimination, the image captured by the camera is transferred from RGB (red, green, blue) color space to HSI (hue, saturation, illumination) color space in this thread.

Considering different designers' working spaces may contain different background colors; and the colors of their physical models also differ from each other. The process is designed to be able to eliminate up to five different sections of the color space (colors background mainly contain) and also force to remain the same amount of sections (colors models mainly contain).

Figure 4.1.2 shows a case which has white models and black background. In that case, designers can set the transparent color to black and the non-transparent color to white. The image after the chroma key process can be found in latter session. Although black and white are in fact the same color with different brightness, the system is still able to eliminate the background fine with the difference of the brightness. The background elimination will work even better when the difference of hue or saturation is larger.

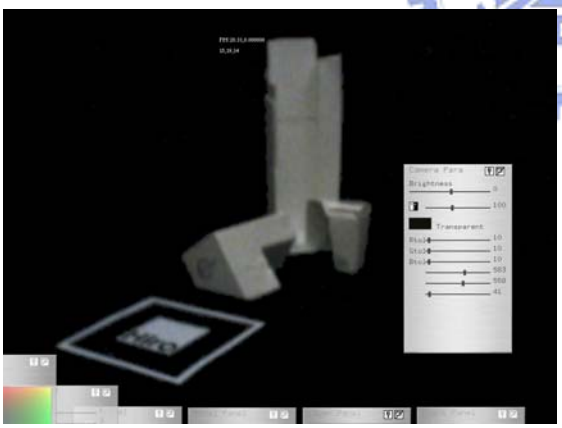

Figure 4.1.2: Designers can set colors to remain or eliminate.

Figure 4.1.3 shows another example for background elimination. In this example, the designer uses a green cutting mat while constructing the physical model. In this commonly used design environment, the system can eliminates the cutting mat completely without affect the physical models.

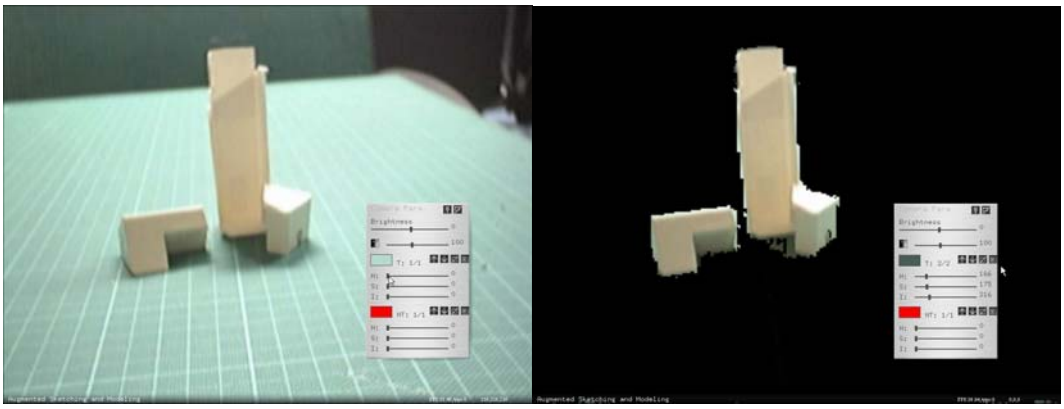

Figure 4.1.3: Example of the background elimination while designer uses a cutting mat in his/her studio.

A panel is constructed to let designers to manipulate these colors. Two different groups of colors can be set with this panel. Each color has three sliders while each slider corresponds to the tolerance for an axis in HSI color space. Figure 4.1.4 shows this panel. The first slider on the panel is for user to adjust the brightness of the images. The button under the brightness slider is for designers to change the camera images into the threshold mode. In the threshold mode the image will be turned into black and white according to the threshold value, which can be set by using the slider next to the button. Threshold mode shows the image that ARToolKit uses for detecting fiducial marker. Due to the difference between different environments, users might need to adjust the threshold value to let ARToolKit detect the fiducial marker correctly. The threshold mode provides a clue for users while adjusting the threshold value.

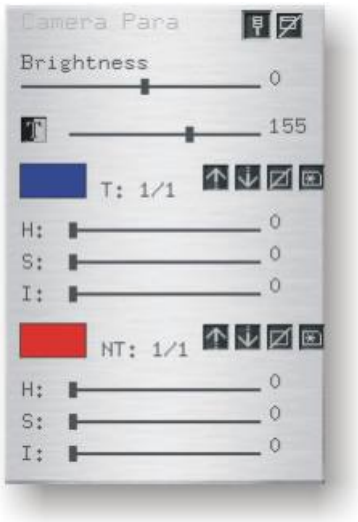

Figure 4.1.4: The panel for configuring camera parameters.

4.1.3 Camera data and the synchronization of models

The fiducial marker, a square marker pattern that is preloaded and learnt by the system, is used in the camera synchronizing thread. The system adopts ARToolKit to analyze the fiducial marker in the video captured by the web camera and calculates the coordinates of the physical camera relatives to physical markers in real-time mode (Figure 4.1.5).

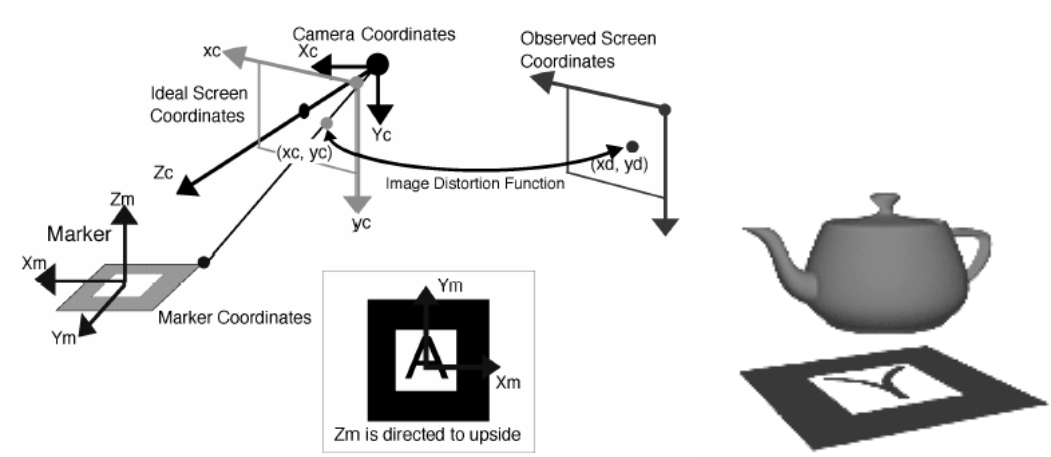

Figure 4.1.5: The basic concept of ARToolKit (after Kato,1999).

The camera synchronizing thread starts when an image is grabbed by the camera. During this thread, the system examines the image to see if the pattern is in it. If more than one pattern is founded, the system uses the one with the highest ranking. The system then calculates the coordinates of the camera according to that pattern. That datum will be transformed into an array containing 16 GLdouble, which is the coordinates datum used in OpenGL.

This research extracts this part of the functionality from the code of the original ARToolKit. A Boolean variable, ”checkPattern”, is added into the code for this system to determine whether to detect the pattern or not considering two kinds of behavior designers might have. One of the behaviors designers might have is to constantly changing the position and orientation of the camera to examine the design with the digital site. This should work with the “checkPattern” on. After doing that, designers would find some specific angles that they are interested in and stop manipulating the camera to focus on the media. This means while designers manipulating the media, the position and the orientation of the camera would not need to be changed. Moreover, since that the fiducial marker is put by the side of the physical media, designers will easily affect the detection of the pattern incautiously while manipulating physical media. This is the situation that the users don’t want the system to keep tracking the pattern. Being able to turn the detection off while designers stop moving the camera also helps the stability of the system. The adapted code is listed below.

```
...
// Grab a video frame.
if((image = arVideoGetImage()) != NULL) {
        gARTImage = image;
        gMainCountCallsMarkerDetect++; // Increment ARToolKit FPS counter.
        if(checkPattern){
                // Detect the markers in the video frame.
                if(arDetectMarker(gARTImage, gARTThreshhold, &marker_info, &marker_num) < 0) {
                        exit(-1);
                }
                // Check through the marker_info array for highest confidence
                // visible marker matching our preferred pattern.
                k = -1;
                for (j = 0; j < marker_num; j++) {
                if (marker_info[j].id == gPatt_id) {
                        if (k == -1) k = j; // First marker detected.
                        else if(marker_info[j].cf > marker_info[k].cf) k = j;
                        // Higher confidence marker detected.
                }
        }
        if(k != -1) {
                // Get the transformation between the marker and the camera into gPatt_trans.
                arGetTransMat(&(marker_info[k]), gPatt_centre,
                gPatt_width, tempPatt_trans);
                gPatt_found = TRUE;
                } else {
                        gPatt_found = FALSE;
                }
        }
        arVideoCapNext();
        while(!imageGrabbed){
        imageGrabbed = true;
}
```

Since the datum of the camera is used for synchronizing, it should also be stored in each canvas. The structure for canvas is extended in this step. An array of GLdouble is added into it to contain the camera coordinates.

```
struct ASMCanvas{
        int type;
        int ID;
        BOOL visible;
        GLdouble    camera[16];
};
```

A 3DS file loader is made to import and decode the digital models. In order to combine the digital and physical models together correctly, a panel is implemented to let designers set the scales of the physical models. The system will render the digital models according to the scale of physical models and the pattern. Figure 4.1.6 shows this panel, the first slider indicates the scale of the physical models and the second one indicates the scale of the pattern. The depth offset slider indicates the offset of the clipping plan for 3D rendering. Designers can use this slider to avoid the situation that the digital models near the camera block the view of the camera. Using the next section of this panel, designers can choose the source of the camera coordinates. Other sliders in this panel are for designers to manipulate the coordinates of the fiducial marker.

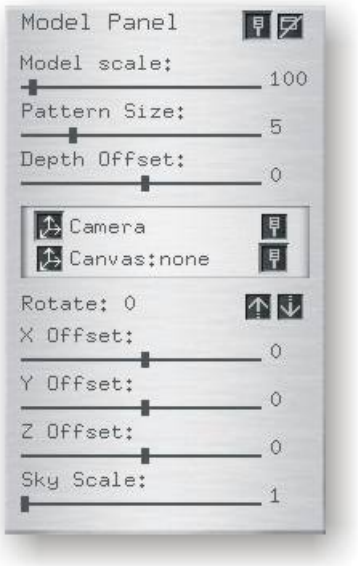

Figure 4.1.6: The panel for setting scales of physical models and pattern.

The system also provides a more intuitive way for designers to manipulate the coordinates of the fiducial marker. Except from viewing the digital site model from the view of the physical camera, this research also allows designers to view the digital models in the "coordinates setting mode". In this mode, designers view the digital models from the top. Figure 4.1.7 shows the interface of the coordinates setting mode.

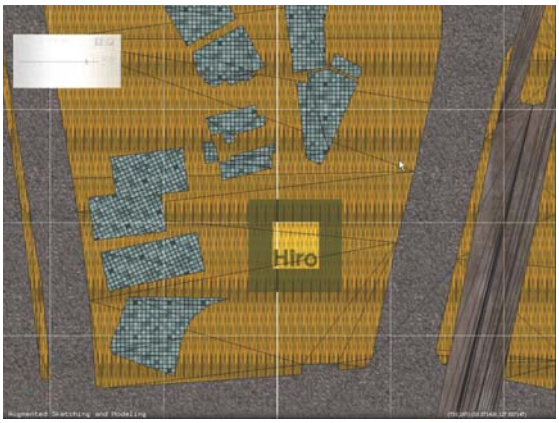

Figure 4.1.7: Designers set the coordinate of fiducial marker in the “coordinates setting mode”

In order to let designer get better control in the “coordinates setting mode”, a panel for controlling the scale of the view is also developed in this step (Figure 4.1.8).

Figure 4.1.8: The panel for controlling the scale of the view.

4.1.4 Depth sorting of digital and physical models

Considering designers might use digital and physical models in different ways. For example, designers might use a physical or a digital site model. Different from normal AR applications, this system should not only be able to add digital models into physical environment but also able to work inverse. Therefore, the system cannot combine these two videos just by putting the video of digital models above the video of physical models. Otherwise sometimes object in the back will covers the one in front (Figure 4.1.9).The problem is considered ignorable in other AR system. However, the spatial relationship between objects is important in architectural design. Therefore, this research has to solve the depth sorting problem.

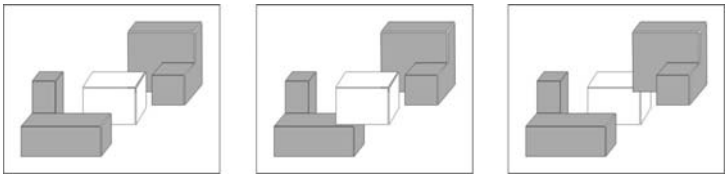

Figure 4.1.9: Image on the left shows the correct image. The other two shows the error occurs when not combining images right.

Depth sorting method 1 to be considered

In order to solve the depth sorting problem, this research first tried to split the digital models into two canvases and renders them separately. First, the scene behind the physical model is rendered, and then the scene in front of the physical model (Figure 4.1.10). This research uses view clipping to achieve this.

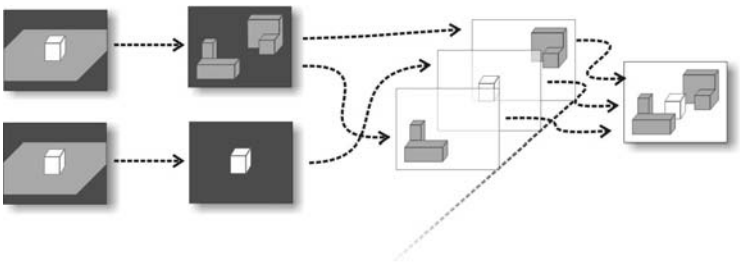

Figure 4.1.10: The system renders digital models twice to make the combined image correct

The following code shows how the system split the digital model. In the code, m[14] means the distance between the camera and the pattern. The system can clip the digital models using this value.

```
...
arglCameraView(gPatt_trans, m, 1);
gPatt_depth = m[14];
...
arglCameraFrustum(&gARTCparam, gPatt_depth, VIEW_DISTANCE_MAX, m2);
...
glMatrixMode(GL_MODELVIEW);
glLoadMatrixd(m2);
glCallList(siteModel);
...
arglCameraFrustum(&gARTCparam, VIEW_DISTANCE_MIN, gPatt_depth, m3);
glLoadMatrixd(m3);
```

```
glCallList(siteModel);
...
```

After the rendering processes of digital and physical models are prepared, the system then render them as individual layers according to their distances to the camera. Designers can now easily modify or change physical models while the system allows for visualizing the image with both physical and digital models at arbitrary angles (Figure 4.1.11). In the combined image (the one on the right) we can see that the spatial order of the models is correct. Digital models that are in front of physical models cover the physical models while those behind physical models are covered by physical models.

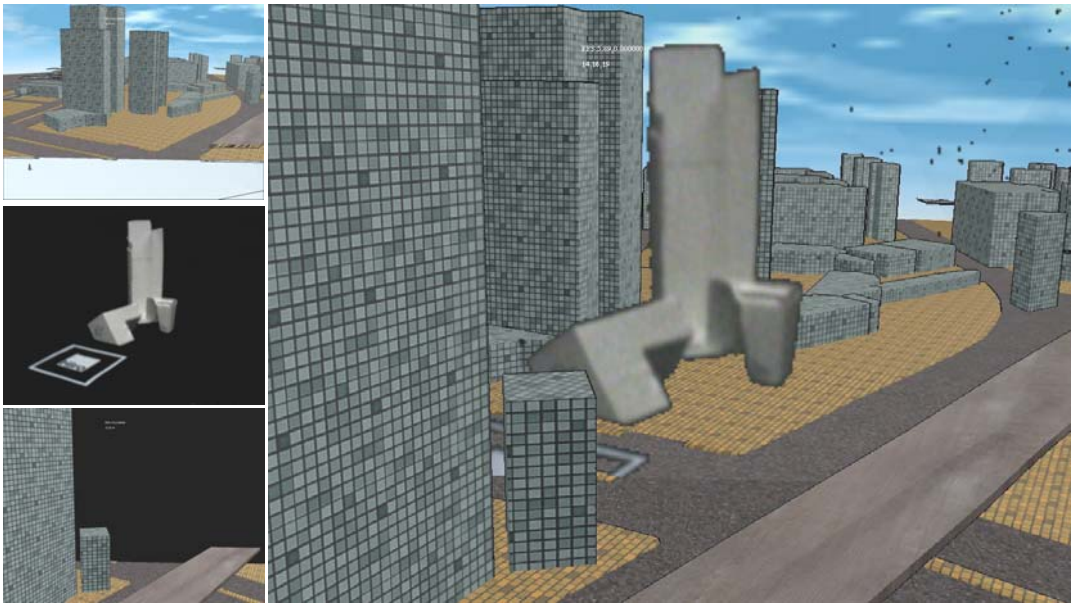

Figure 4.1.11: Images on the left are digital models and physical models. On the right is the combination.

Figure 4.1.12 shows images of designers changing different views and models. The system updates the change made by designers in real-time.

Figure 4.1.12: The output images of different views and models.

However, since this method use only one single depth value to clip the digital models, there are errors in some situations. Figure 4.1.13 shows two situations that cause errors with depth sorting. Since the physical models have variance depth values for each part, the depth value for cutting digital models should always be the shortest distance between the camera and the physical models. This means it is not possible for the system to find the correct depth value while knowing only the distance between camera and the fiducial marker. Otherwise the situation like image on the left will happen. In this case, the fiducial marker is at the side of the physical model. Which cause the detected depth value is too deep that cuts off parts of the physical models.

Even if the system gets the correct depth value every time, there is another problem that might occur. Image on the right shows another situation that causes an error. In this case, the digital building next to the physical models should be in front of the physical models. However, in order to avoid cutting the physical models, the depth value becomes too near and results the wrong depth sorting.

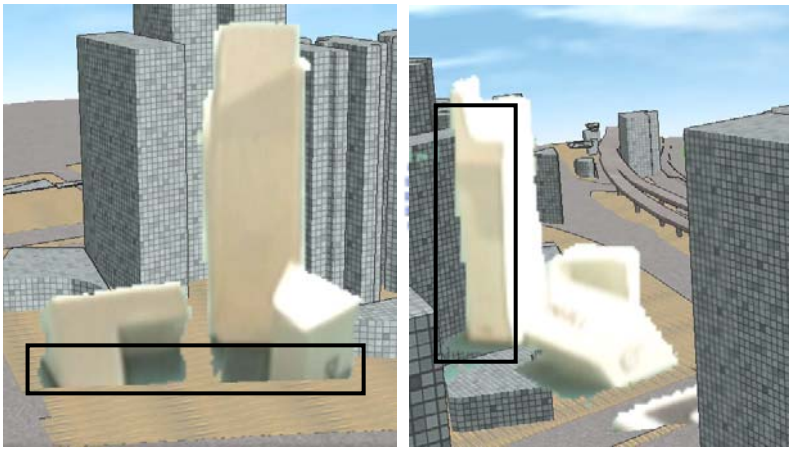

Figure 4.1.13: Some situations that cause errors using this depth sorting method.

In conclusion, this depth sorting method can creates correct depth sorting images most of the time but would be wrong in certain situations. To sort the models correctly, more information about physical models is needed. Since the spatial order between models is important to architectural design, this research tries another approach to solve the depth sorting problem.

Depth sorting method 2 used in the system

The result of the first attempt shows that more information of physical models is needed for generation of the correct depth value. Moreover, each pixels of the physical model image should have different depth value, which is just like the situation of the real physical model.

In order to make the 2D physical model images become 3-dimentional, this research uses a “physical model volume” to map the 2D images into 3D space. The “physical model volume” is a digital 3D object that is generated after designers mark the ambit of the site. Figure 4.1.14 shows the interface for designers to set the boundary of the site. Four points and a height should be defined to construct a volume. Designers are able to setup several volumes in order to create a site with a complex shape.

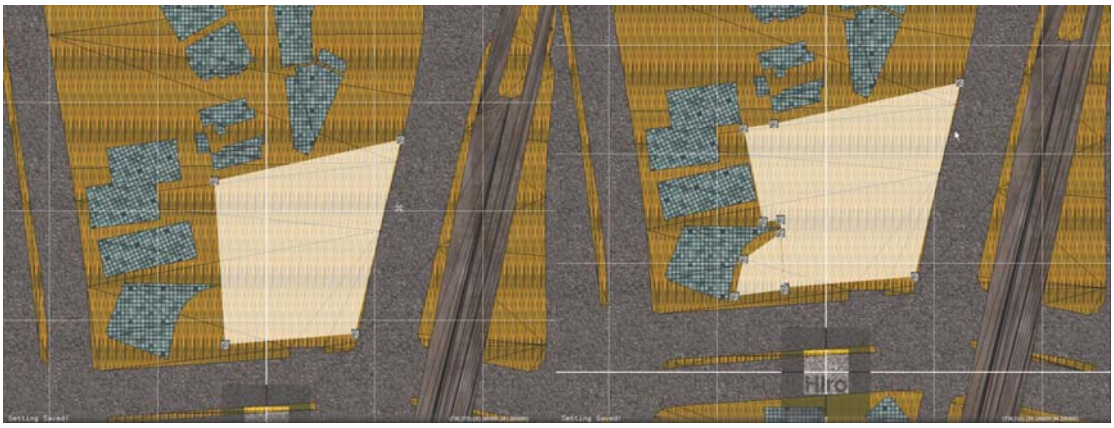

Figure 4.1.14: Interface for designers to set the boundary of the site.

While the boundary setting interface is designed to be in the coordinates setting mode, a panel is developed for designers to set the height of the physical model volume (Figure 4.1.15).

Figure 4.1.15: The panel for designers to control the height of the physical model volume

With this “physical model volume”, the system renders the volume with the digital model from the camera before the original rendering process. While doing this render, the system disabled the texture mapping and the lighting. The color of digital models is set to black and the color of physical model volume is set to white. Figure 4.1.16 shows the result of this rendering. Image on the left is rendered with texture mapping and lighting. The one on the right is the render with black digital models and white physical model volumes.

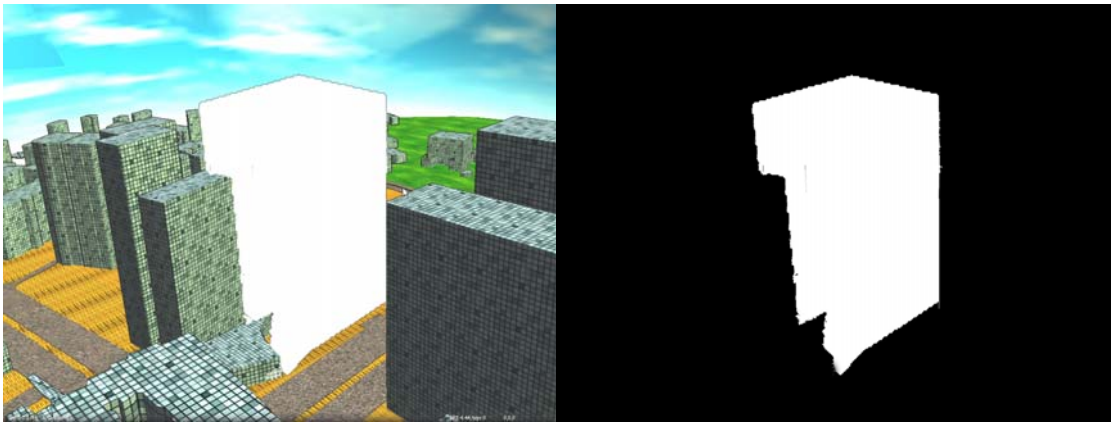

Figure 4.1.16: The rendering image for depth sorting process.

This black and white image shows the region where physical models should be visible. Due to the physical model volume is a 3D object; the system can do the depth sorting directly using OpenGL. In the left image of figure 4.1.16 we can see that the spatial order of the digital models and physical model volume is correct. The system then uses this image as an alpha testing map on the images of physical models. This also helps the background elimination process. In figure 4.1.17, image on the left shows the image being cut using the alpha testing map.

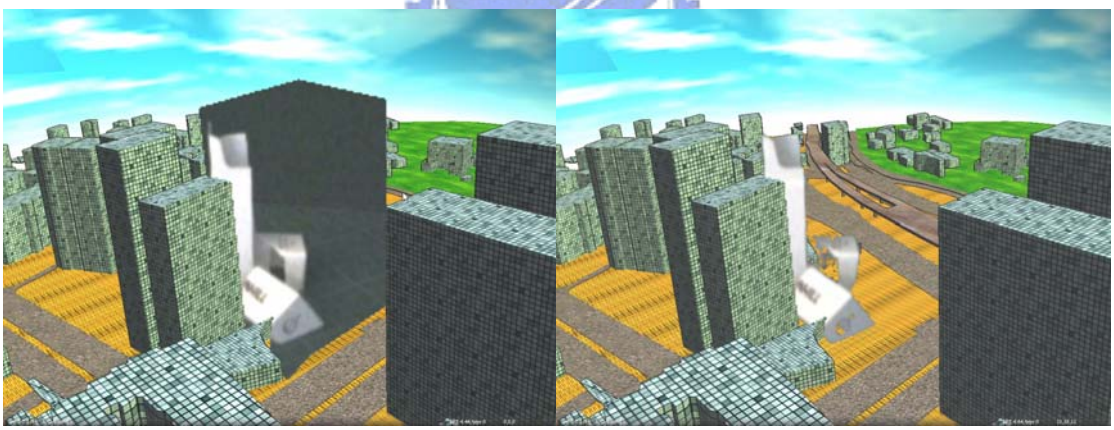

Figure 4.1.17: Image on the left shows the image of physical models mapped on the physical model volume. Image on the right shows the result of the depth sorting process.

The original background elimination process comes after the alpha cutting. It eliminates the parts that are not the model and complete the integrating of physical and digital models. Figure 4.1.18 shows the output images using this depth sorting method. Two images on the left are the error images using the previous method. The next two images are the result using the “physical model volume”. We can see that the problems are solved by using the second method. With “physical model volume”, the system now can combine images of digital and physical models with no error.

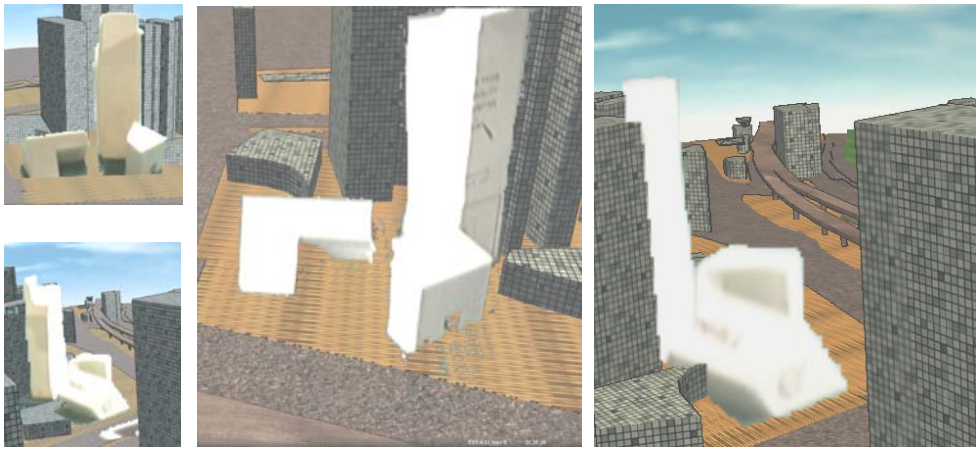

Figure 4.1.18: The two images on the left show the result using original depth sorting method, which causes error. The other two images are the result using the second depth sorting method, which creates correct images.

4.1.5 Using digital and physical models together

As described earlier, designers can construct and manipulate physical models while using this system. Since the camera keeps capturing the images of the physical models, every changes made by designers will be updated in real-time mode. As we can see in figure 4.1.19, designers can put the physical models inside the digital environment and mark on it. This behavior is just as what designers would do while having another physical models on the side. The benefit is that designers will not have to reconstruct these models while switching within media. Designers can construct the part by any medium they want, and it will keep assist designers even if designers have already switched to another medium.

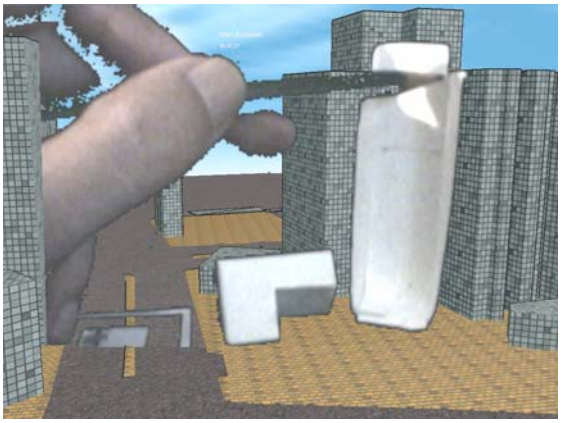

Figure 4.1.19: Designers can put the physical cardboard inside the digital environment and manipulate it.

Moreover, not only physical models but also other physical tools can be blended into digital environment. This means many common used physical tools, such as a ruler, become tangible devices for digital world. In figure 4.1.20, we can see that designer uses a physical scale to measure the height of a building in digital model. Also, sketches on physical papers can also be integrated in this way. This will be described in next session.

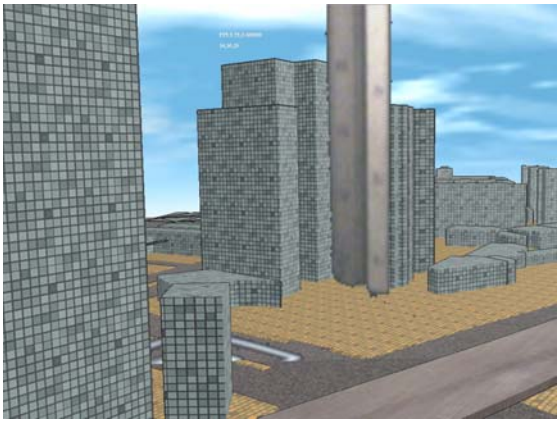

Figure 4.1.20: Designer uses a scale to measure the height of a digital model.

4.2 Sketches in the system

The combining of sketches is different from combining models or other media. Since the output of this system is 2D images, which is compatible to sketches; the sketches in the system are designed to be combined with the final output images. A tablet monitor is used to achieve this attempt. This tablet monitor is the display of the final result and also the device for receiving the input of the sketches. In this step this research implements the functions for receiving inputs from the tablet monitor and for integrating sketches with other media.

4.2.1 2D digital sketches using tablet monitors

Due to the tablet monitor, unlike normal mouse or keyboard, is a kind of advanced graphic input devices; a receiver needs to be constructed in order to let the system receive and decode the data from the tablet monitor. This system uses Wintab API to establish this receiver. This receiver is able to receive and decode data from tablets and tablet monitors that are most commonly used. This research then uses the GLUT mouse handler to pass the data into the system. Using the tablet monitor, designers can sketch directly on the video that already incorporates both the physical and the digital models. The code below shows how the system adds tablet information in the original mouse handler.

```
void processMouseMotion(int x, int y) {
```

```
            AppGL.MyProcessMouseAMotion(x,y);
            td.getNextEvent(evt);
            GLPoint->tabletPressure = evt.pressure;
            GLPoint->tabletType = evt.type;
    }
```

There are two kinds of tablet data: “pressure” and “type”. The “type datum” shows it is the head or the bottom of the pen designers are using. The system interprets this datum as a pen brush or an eraser. This makes designers use the head of the pen to write and the bottom to erase; just as using an ordinary pencil. The “pressure datum” is a float between 0.0~1.0 which indicates the pressure of the pen. The system uses two integers to translate the pressure into the pen brush on the image. As shown in figure 4.2.1, this panel constructed in this step contains two sliders corresponding to each integer. These integers indicate the size of the pen brush when the pressure is 0.0 or 1.0.

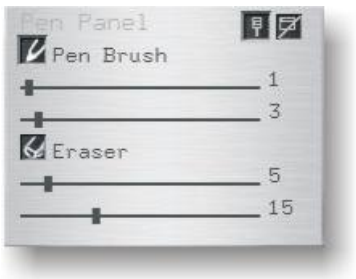

Figure 4.2.1: The panel for configuring size of the pen brush.

Also, a panel for users to select colors is also implemented in this step. Users can use this panel to select the color of the pen brush. Also, the panel can be used for selecting the color for transparency setting (Figure 4.2.2).

Figure 4.2.2: The panel for configuring color of pen brush.

Different from data of models, which are designed to be stored in particular canvases, data of sketches is designed to be able to store in many different canvases. In order to enhance the connectivity between sketches and models, designers should be able to directly modify the images of the physical models. However, the original canvases of physical and digital models data are continually refreshed by the incoming images. Every changes made by designers would be cleared by the system. Therefore, to let designers sketch on the physical model images, function that duplicate canvas is added into the system in this step (Figure 4.2.3).

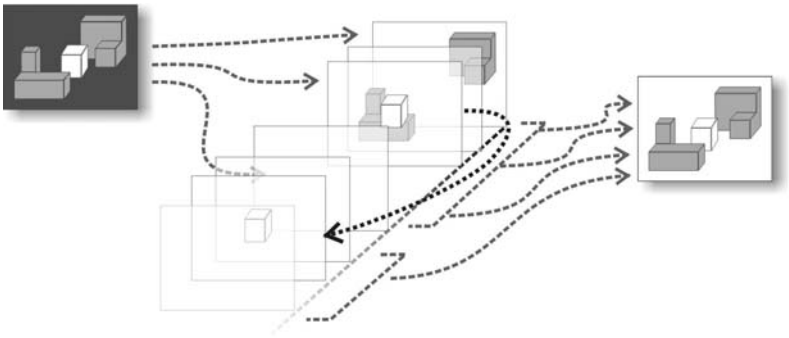

Figure 4.2.3: While other media have their specific layers, sketches uses all the other layers. The source of sketches can be directly from designers or just duplicate the layers of other media.

Designers can duplicate the static image of the models and directly sketch on that image, which may be a reference for future models. To let designers directly sketches on the duplicated canvases, the system transforms the original canvas into default canvas (the one using MyCanvas) while duplication the canvas. When designers duplicate a canvas, the system will start to render the duplicated physical model canvas using frame-buffer before the display rendering process. Then the image in the frame-buffer is copied to the destination MyCanvas.

4.2.2 Sketches on models

As we can see in figure 4.2.4, designers can sketch on the image that contains both physical and digital models. Designers can choose to sketch on a blank canvas or a duplicated medium canvas. No matter which kinds of canvas designers choose to use, the sketching will be augmented with the information of other media.

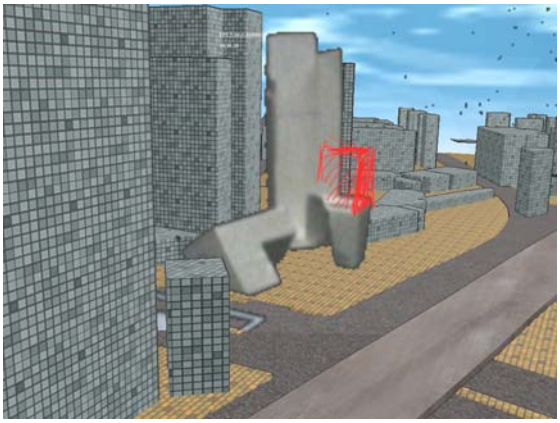

Figure 4.2.4: Designers sketch on the image that contains both physical and digital models.

Also, designers can combine different parts of different physical models by duplicating images of physical models (Figure 4.2.5). In this case, the designer first duplicates the image of a physical model into a sketching canvas. Then the designer erases part of the image and replaces the physical model with another physical model. By turning the camera canvas back on, the system automatically combines two physical models together.

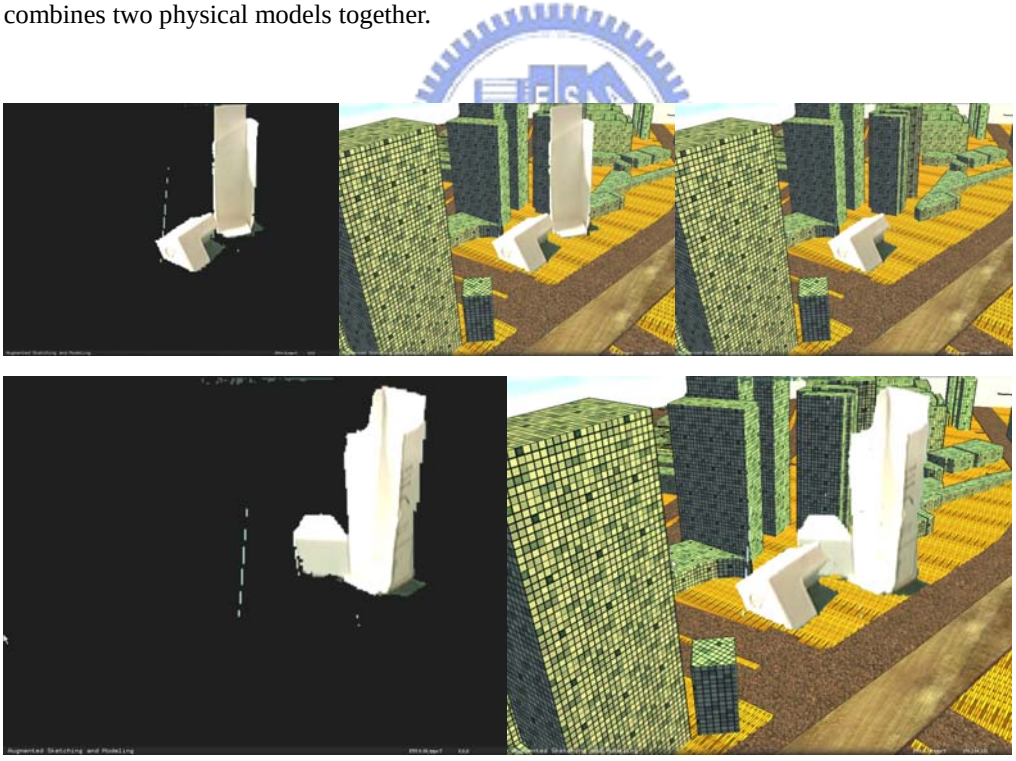

Figure 4.2.5: Designers can combine different part of various physical models by manipulating the duplicated canvases.

Besides models, other physical tools can also be used to assist the sketching. The left image of Figure 4.2.6 shows the image grabbed by camera while designers use a right angle. In the image on the right,

the designer sketches the correct 2D perspective line with the help of the right angle in the 3D physical world. The system use the camera to transform 3D objects into 2D coordination and helps designers to sketch perspective drafts correctly.

Figure 4.2.6: The image on the left is captured by the camera while designers using a right angle. Seeing the right angle while sketching can help designers easily sketch the correct perspective line (the image on the right).

Since the coordinate of the camera has already been stored in the canvases, these data will also be copied into new canvas. Designers are able to synchronize the camera of the digital models with the canvas. This helps designers to efficiently browse and modify the idea they generated for different angles.

4.2.3. Integrating 2D sketches in digital and real world

Since all physical objects can be integrated into the digital environment, sketches on the physical papers are also combined into the system. Designers can sketch both physically and digitally using the system. As we can see in figure 4.2.7, designers can sketch on a paper while seeing that paper in the digital world.

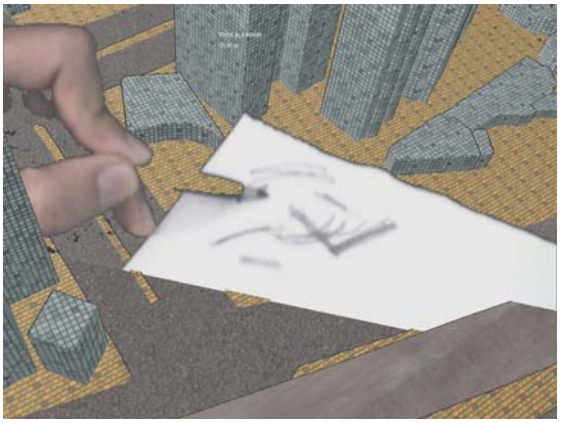

Figure 4.2.7: Designers sketch on a paper while seeing the paper put in the digital world.

After that, designers can sketch a perspective draft according to the layout plan they just sketched on the paper (Figure 4.2.8). Since the plan is on a physical paper, designers can change the view of the camera to any angle and use the plan to assist drawing perspective drafts.

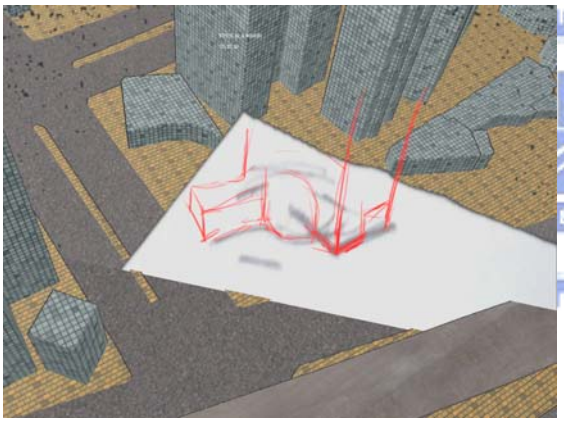

Figure 4.2.8: Designers sketch a perspective on the tablet according to the sketch on the paper.

As described earlier, the camera helps to transform 3D objects into 2D coordination to help designers sketch. On the opposite, the camera also helps to transform 2D images into 3D coordination. The 2D sketch can also be a reference for making physical models. As we can see in figure 4.2.9, designers use the sketch to mark the shape on the cardboard. Seeing the 2D sketches projected on the cardboard, it will be easier for designers to make the physical models since the system did the coordination transformation for them.

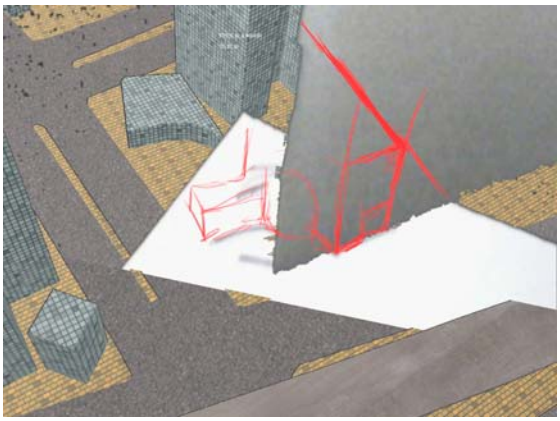

Figure 4.2.9: This sketch can also be a reference for physical models. In this image, designers combine a cardboard with the sketch to mark the shape on the cardboard.

These sketches are only 2D sketches since the system only combines traditional sketches at this stage. However, other new sketching media such as 3D sketches can be added into the system as a new medium in future works.

4.3 Adding new elements on digital models

In this part of the step, the research develops a digital medium and adds it on digital models. Considering the two major purpose of this part of the step: to demonstrate the extensibility of the system and to observe the result of providing more visual information to designers. This research chooses a simple medium, which is a dynamic animated site model that provides more environment information. Figure 4.3.1 shows the parts added in this part of the step. If any other digital media are to be combined into the system, repeat and modify this step will do the work.

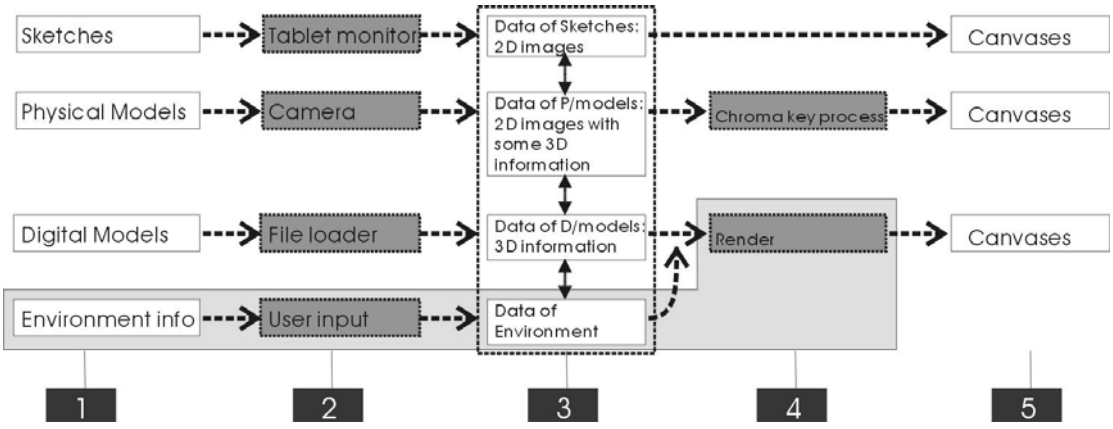

Figure 4.3.1: (1) media data (2) media data receive process (3) media data analyze and synchronize (4) media data transform (5) data storage (canvases)

The additional information implemented in this digital media contains traffic condition of nearby environment, paths of nearby pedestrians, and shadows cast by nearby buildings. This research chooses this information as the example because this information is considered important in architectural design. These information objects are added on the digital models. Since the output of digital models has already been combined with physical models and sketches, designers will receive this additional information while using physical media as well. The user interface of additional information is also extended from the GUI prototype.

4.3.1. Activities information

The activity on the site is always an important thing to be considered in architectural design. However, this information often appears only in an abstract form. This research magnifies the feedback of activity information by transforming the abstract information into visible form and combines it with other media. This should provide more visual feedbacks to designers and affects the design.

The activities information in this research includes the traffic condition and the paths of the pedestrians. It is displayed as real-time rendered animations and can be set and modified. Besides of the database for storing the activities information, the one stores data for transforming information into digital models is also implemented in this step. Some basic animated digital models such as the models of cars and walking pedestrians are constructed and loaded into this database before designers start to use the system. Moreover, an independence thread is also implemented in this step. This thread starts and processes rapidly once designers setup the additional information. It transfers the data into animated 3D objects and imports these 3D objects into digital models. Figure 4.3.2 illustrates how these two data storages and the thread work. The information database stores the properties of the activities while the model database stores the digital model for transforming activity information to animated models. Users manipulate the information database through the user interface and the thread uses these data and the preloaded model data to create visible scene in the digital models.

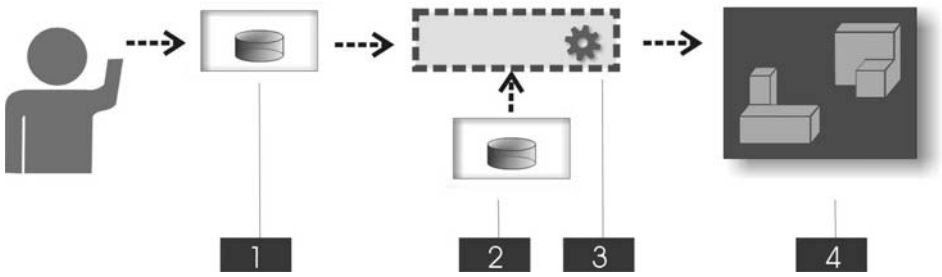

Figure 4.3.2: (1) Activity information database. (2) Animated model database. (3) Transforming thread. (4) Digital model output.

Information such as traffic condition and pedestrians is based on the actual situation on the site. For example, designers may setup traffic information with the traffic flow of the road in front of the site. Figure 4.3.3 shows the interface for manipulating activities information. The system also uses the coordinates setting mode as the interface for setting activities information. To let designers set the activity information, a panel is also constructed.

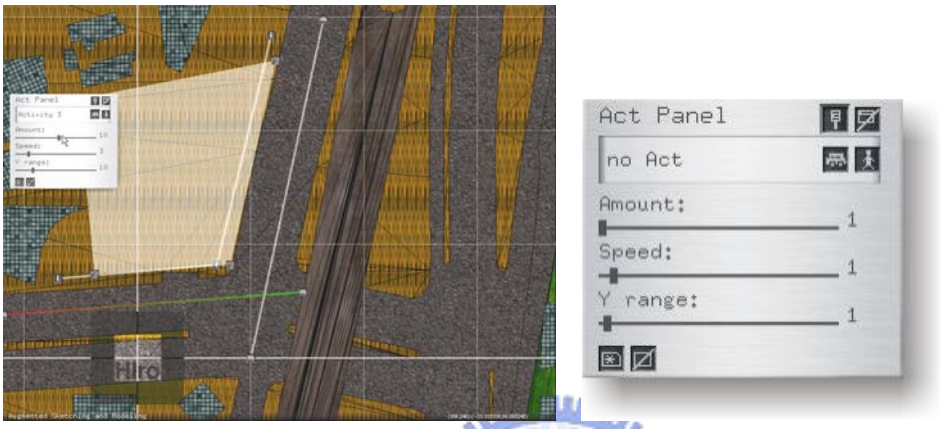

Figure 4.3.3: interface of moving the coordinates of activity objects

The user set a traffic flow information object in front of the site. Figure 4.3.4 shows the images of digital model with and without the activity objects.

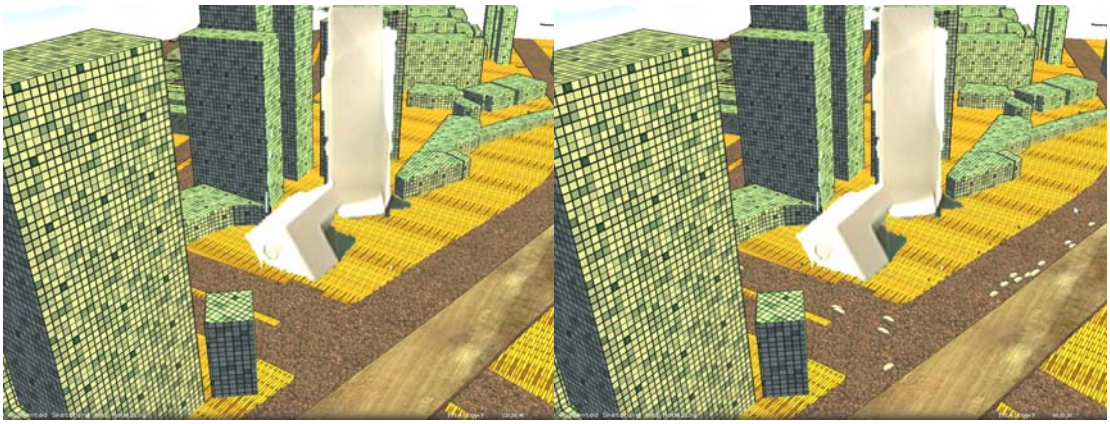

Figure 4.3.4: Images of the system before and after adding the activity information

When the activity information object is created, a thread mentioned in the last session is also created. This thread calculates the amount of the cars and the position of the cars by the data inputted by designers. Also, the thread keeps rapidly controls the animation of digital model by refreshing position

of the cars according to those traffic data. Once the information of traffics is calculated, the thread transforms the information into digital models by using preloaded models. Each activities information object is independent to each other; every datum designers input has an individual thread. The system is designed to allow designers to enter up to twenty “activity objects” like this to simulate the actual situation on the site.

4.3.2. Shadow information

Except from activity information, shadow cast by nearby buildings is another important thing to be considered in evaluation. Although shadow and sunlight deeply affect the result of the design, the evaluation with them often begins only in the latter step of design process. Therefore, this paper considers shadow as an important element to be added into visual feedback in the conceptual stage of design.

Shadow information is a little different and more complicated comparing to activity information. What should be input by designers are the location of the site, the time and the date that would be simulated. The process transforms these data into the main directional light source in the digital models. In this step, this research makes some changes to the way the system renders the digital models. An algorithm for calculating shadow is added into the rendering process. Considering the calculation speed, the amount of lights, accuracy, and the changing rate of the shadow. The system uses shadow mapping algorithm for shadow calculation. Shadow mapping, presented by Williams (1978), is a complete image-space algorithm which does not need to do the geometry calculation.

This research creates a thread to implement shadow mapping. The thread starts when the light source or digital models is changed. Before digital models are rendered from the view of the camera, the thread renders digital models from the light’s point of view. Using the depth buffer (z-buffer) of GPU (graphic processing unit), the thread can generate a shadow map by projecting digital models onto the light source plane. The shadow map shows the depth (distance) of the points in 3D objects that is projected onto the light source plane; which is the nearest point in all points that are projected onto the same pixel in light source plane. After the thread generated the shadow map, the comparison between distance of point projected on the light source plane and the distance stored in shadow map is made while rendering digital models. If the distance between that point and the light source plane is longer than the distance stored in the shadow map, the point is shadowed. On contrary, if that distance equals to the shortest distance, the point is not shaded (Figure 4.3.5).

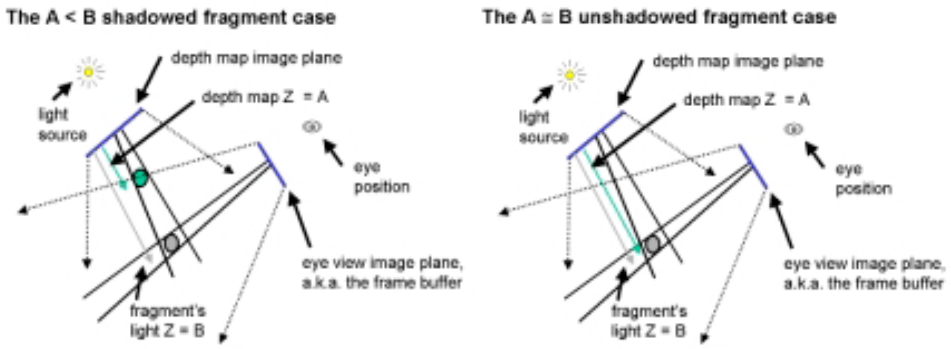

Figure 4.3.5: The main concept of shadow mapping

The system uses this efficient algorithm to calculate the shadow in the 3D scene in real-time mode. This helps designers to get well known of the situation how nearby buildings cover the sun on any date and in any time.

Designers can set the parameters about the sun with the panel constructed in this step. Figure 4.3.6 shows this panel. With this panel, designers can set the latitude of the site, the date, and the time. The time scale slider is used for setting the animation. With the time scale set to zero, the animation stops. Larger time scale indicates the shadow animation goes faster. Sun distance slider is for designers to set the distance of the sun to fit the size of the digital model. Sun width slider indicates the area that the shadowmap covers. Since the detail of the shadow depends on the size of the shadowmap, designers can use this slider to change the area that shadowmap covers to get the better details.

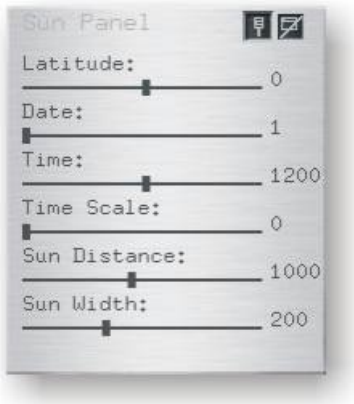

Figure 4.3.6: The panel for designers to adjust the sun parameters

The content of the system canvas database is changed again in this step. Date and time are also designed to be stored into each layer. The purpose of doing this is as same as storing the data of the camera angles. Designers can quickly switch within ideas when they considered different angles, time,

and date. After this step is done, the system combines sketches, physical models, digital models, and environment information (Figure 4.3.7).

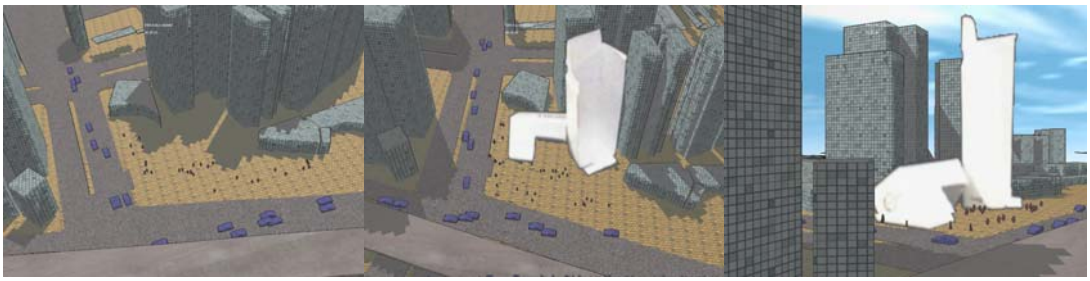

Figure 4.3.7: The images of media combined with each other.