Chapter 5

Design and Implementation

This Chapter discusses the implementation of the Expressive Texture theoretical approach described in chapter 3. An avatar creation tool and an interactive virtual pub environment for these avatars are implemented. The interactive virtual environment simulates a real-world social environment, which allows the user to observe the social interaction between avatars, and the emotional changes of avatars caused by the social interaction. Section 5.1 discusses the implementation and section 5.2 discusses the creation process of the virtual pub environment. Section 5.3 discusses the modification of the expressive texture tool into the avatar creation tool. Later in Section 5.4, the implementation of the social interaction application will be discussed and determined how the avatar creation tool was linked to it. The chapter is summarised in section 5.5.

5.1. Avatar Generation

An avatar can be generated by importing the prototyped-avatar models from the modelling tools into the application or obtained the geometric data of the prototypes and implement the avatar based on these data. The application was developed using C++ and OpenGL. Because the existing importing plug-ins for does not support complex texture mapping on the models, the avatar models were not importing.

5.1.1. Body Creation

The avatar data model (the position of all vertices) was created in a 3D modelling application. Based on the data of the prototyped avatar, the avatar model was implemented in OpenGL by constructing each body part as a polygon object. Each body part of the avatar model was created using the vertex data of the corresponding body part of the prototyped model. Therefore, the avatar model in OpenGL had the same scale as the prototyped model and the positioning of each body part remained in the correct position as the body parts in the prototyped model. Each body part can be coloured individually. However, the hierarchy structure of the body and the adjusted pivot positions at the joints of the upper limbs were not preserved in the vertex data. Therefore, the body parts are regenerated in a bottom-up manner in OpenGL. The pivot points at the joints between the upper body limbs are also adjusted.

In OpenGL, the hierarchical body structure was created using the Matrix Transformation Stack, the objects were pushed into the matrix stack using a method "glPushMatrix()" and they were popped by calling the method "glPopMatrix()". The Higher level objects (e.g. body) were pushed into the stack before the lower level objects (e.g. arms, legs, etc.) were pushed into the Matrix stack. Because the Matrix stack has LIFO (Last in, first out) properties, the lower level objects were drawn first. Any transformation and rotation applied on the lower level objects will not affected the higher level objects, because the transformation and rotation occurred before the higher level objects were drawn. However, the rotation of the higher-level object causes the lower level objects to rotate as the higher level rotation is applied after the lower level object was drawn.

Apart from ensuring that the correct rotation and transformations are applied to the specific levels of objects, the matrix transformation stack also preserved the root co-ordinate position.

Figure 5.1, shows a simple example of the code of the method drawing the female avatar body using the matrix transformation stack. The structure was based on the tree representation of the female avatar model in the previous chapter.

The male avatar's drawing method is coded similar to the female avatar, but the difference is that the male avatar has a hierarchy coding with "draw_RightThigh()", "draw_RightLowerLeg()", "draw_LeftThigh()" and "draw_LeftLowerLeg()" instead of the "draw_Dress(lower)" method.

```
Public void draw_femaleAvatar(){

    glPushMatrix();
        draw_neck();
        draw_Body(shtGender, top);
        glPushMatrix();
            draw_face(avaNum);
            draw_backHead(avaNum);
        glPopMatrix();
        glPushMatrix();
            draw_RightShoulder(top);
            draw_RightUpperArm();
            glPushMatrix();
                draw_RightLowerArm();
                glPushMatrix();
                    draw_RightHand();
                glPopMatrix();
            glPopMatrix();
        glPopMatrix();
        glPushMatrix();
            draw_LeftShoulder(top);
            draw_LeftUpperArm();
            glPushMatrix();
                    draw_LeftLowerArm();
                    glPushMatrix();
                        draw_LeftHand();
                    glPopMatrix();
                glPopMatrix();
            glPopMatrix();
```

```
                    glPushMartix();
                        draw_Dress(lower);
                        glPushMartix();
                            draw_RightFoot(shoe);
                            draw_LeftFoot(shoe);
                        glPopMatrix();
                    glPopMatrix();


                glPopMatrix();
            glPopMatrix();
}
```

Figure 5.1 A simple method using matrix transformation stack to perform the drawing of the female avatar.

5.1.2. Animating Body motion

When each body part was created in OpenGL, the origin of the local co-ordinate system is initially positioned at the centre of the body part object. When adjusting the pivot of the body limb for animating body motion correctly, appropriate modelling transformations are applied to the body limb that is going to bend at a specific angle. E.g. When animating the lower arm bending, 3 matrix is used to accumulated the translation, rotation and scale of the avatar body parts (body, upper arm and lower arm). The lower arm is animated based on the origin of the avatar, it is translated to the origin of the avatar. Then rotation was applied to the lower arm, which it rotated around the origin of the avatar and rotation is applied to the lower arm, before translated it back to the position connected to the upper arm and draws the lower arm object (Figure 5.2). These translations applied during the animation of lower arm model are apply quickly between frames at run-time before drawing the lower arm, such that the user will not noticed any flickering, when animating the lower arm rotation. This modelling translation simulated the bending of the lower arm and similarly these translations can be applied to the other body limbs to simulated limb movement.

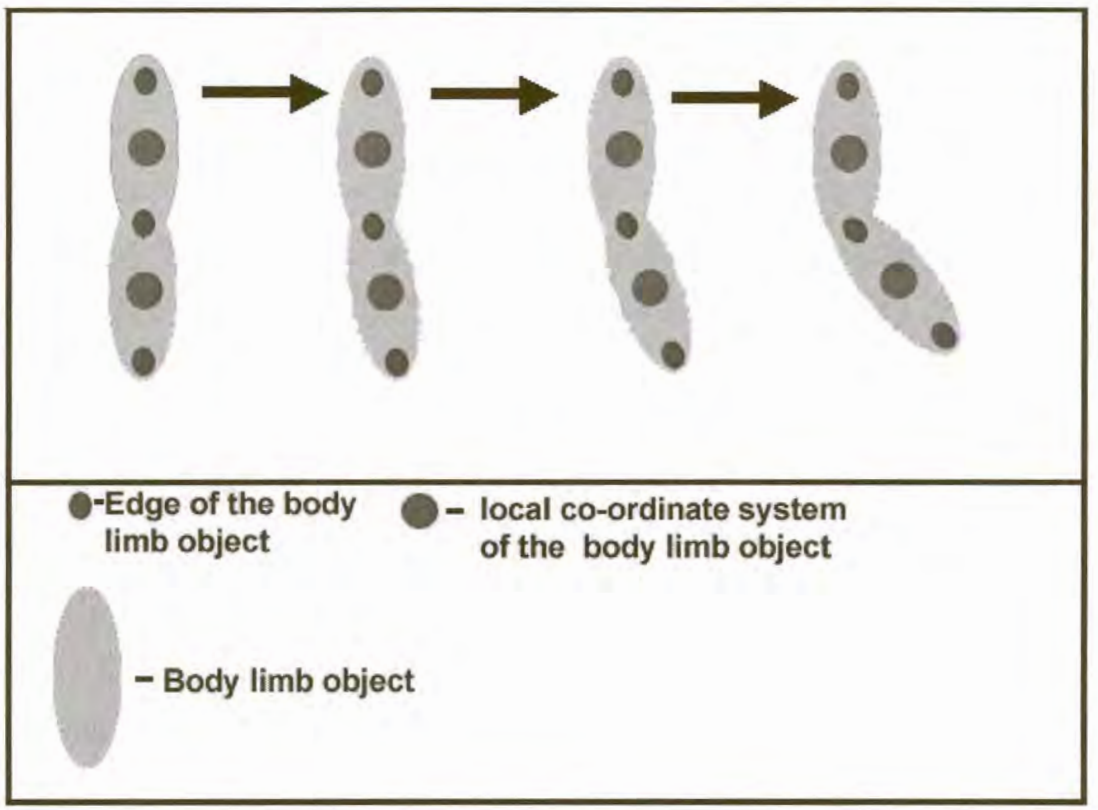

Figure 5.2 The side view of the modelling transformation sequences for a bending body limb.

Once all the modelling transformations are determined for every movable body limb, upper body motion can be animated on the avatar based on the motion cues of the possible body motions of the avatar defined in the previous chapter.

The data containing the amount of upper body movement for each emotion and action are determined and stored in the system. These data are used as the targeted limb motion when calculating body animation, which the limb object incremented its motion values (Pivot angle of the specific limb) from the current limb position to reach the target position during animation. E.g. When animating the lower arm bending, the pivot angle between the lower arm and the upper arm must be increased gradually, starting from zero degrees angle when the lower arm is in a resting position.

The motion value is continuously increased or decreased gradually during animation. It is implemented in a method or function that is updated in every frame. The function call is, glutIdleFunc() in the OpenGL Utility Toolkit. After the calculations and motion

values are coded in the idle function, the body motion of each avatar is animated when an action or emotional event triggers the body animation (Figure5.3, Figure 5.4, and Figure 5.5).

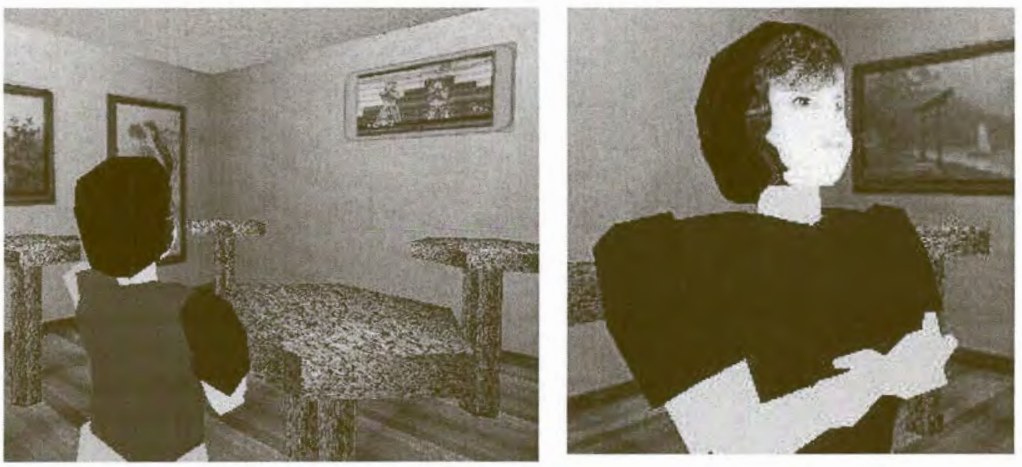

Figure 5.3 The avatar is watching television.

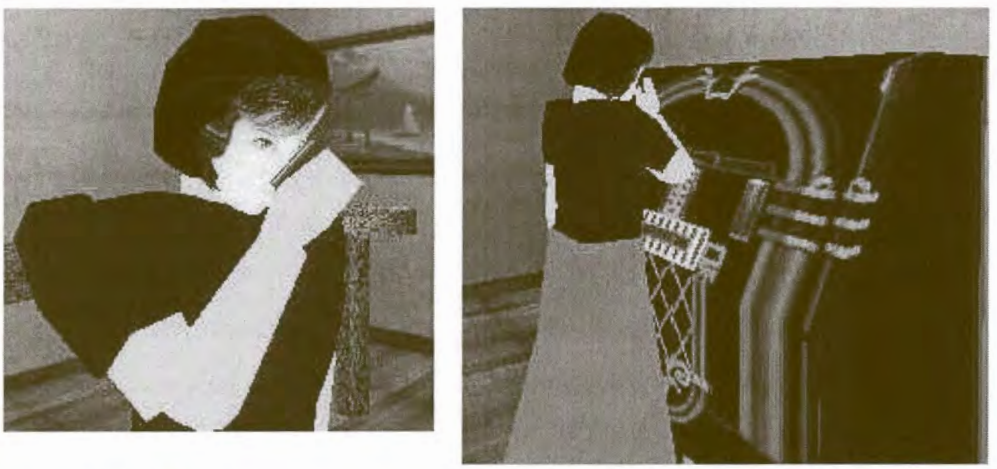

Figure 5.4 The left image shows the avatar drinking a can of beer and the right image the avatar is listening to the music from the jukebox.

Figure 5.5 The avatars talking to each other.

5.2. Virtual Environment Generation

The virtual environment of the avatar must be designed and generated to support the interaction of the avatars. The objects in this environment are designed and drawn in the correct proportions corresponded to the real world objects, so interactions between the avatar and objects in the virtual environment appear natural and fairly realistic from the user's point of view. Similar to the avatar, the objects in the virtual environment are modelled with low polygon number and textured with surface images of real objects.

5.2.1. Virtual Pub Environment and objects creation

The virtual environment and objects can be implemented into the system without prototyping, because the geometry of the virtual environment and objects are simple and do not have the complex hierarchy structure. In order to simplify the system, the virtual environment and objects cannot perform animation and there are no external forces (e.g. virtual wind) affecting the objects in the virtual pub environment, because it

is a closed environment. Therefore, the model of the virtual pub and the objects in the virtual environment are modelled as simple geometric shapes.

5.2.1.1. Virtual Pub Environment

The virtual pub environment is generated as a room-like environment with six rectangular faces that represented the walls, ceiling and floor of the pub. A large rectangular opening is created in one of the wall to represent a door opening. The door was created with the same size as this opening in the wall, which can translated horizontally simulating a sliding door.

In the real environment, people, objects and wall structures were solid and had properties occupied space, which prevented people walking through it. However, these properties are not represented in the objects and wall structures in the virtual environment. Therefore, some form of collusion detection method was required to prevent this problem form appearing during the avatar movement.

Instead of computing the object surface for collusion detection, a simple grid approach was implemented for collusion detection. The floor of the virtual pub environment was sub-divided into a grid structure with square cells that had the width of the avatar (Figure 5.6). The floor of the virtual pub environment is similar to a room plan that contains the position of furniture, the position of the virtual objects and avatars are marked on the cells in the grid. When the avatar walks in the virtual pub environment, it can detected if other avatars or a virtual object occupied the adjacent cells and try to move past that cell without walking through the virtual object or other avatar in that cell.

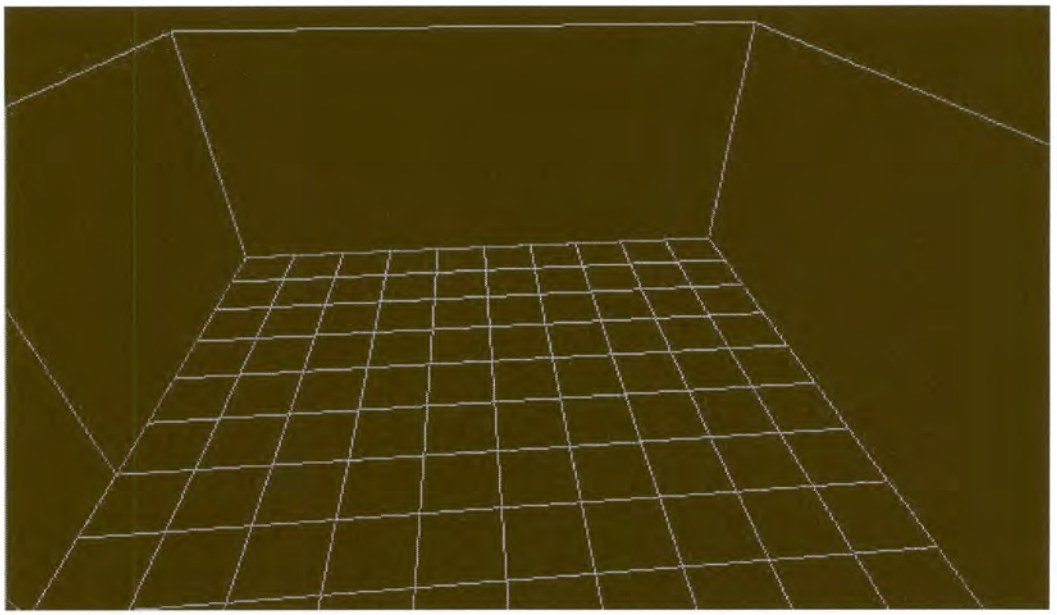

Figure 5.6 The wire-frame view of the floor in the virtual pub environment, which is divided into a grid to detected space availability.

5.2.1.2. Virtual objects

The possible objects in the virtual pub environment are tables, television, and jukebox, bar table, lights and cupboard (Figure 5.7). Instead of drawing all the objects in the virtual environment using a single method and each object were drawn by calling the corresponding method that drawn the object. The advantage of drawing each object using a single method is that the object can be repeated by calling the method again with another transformation that would position or scale the object in a new place.

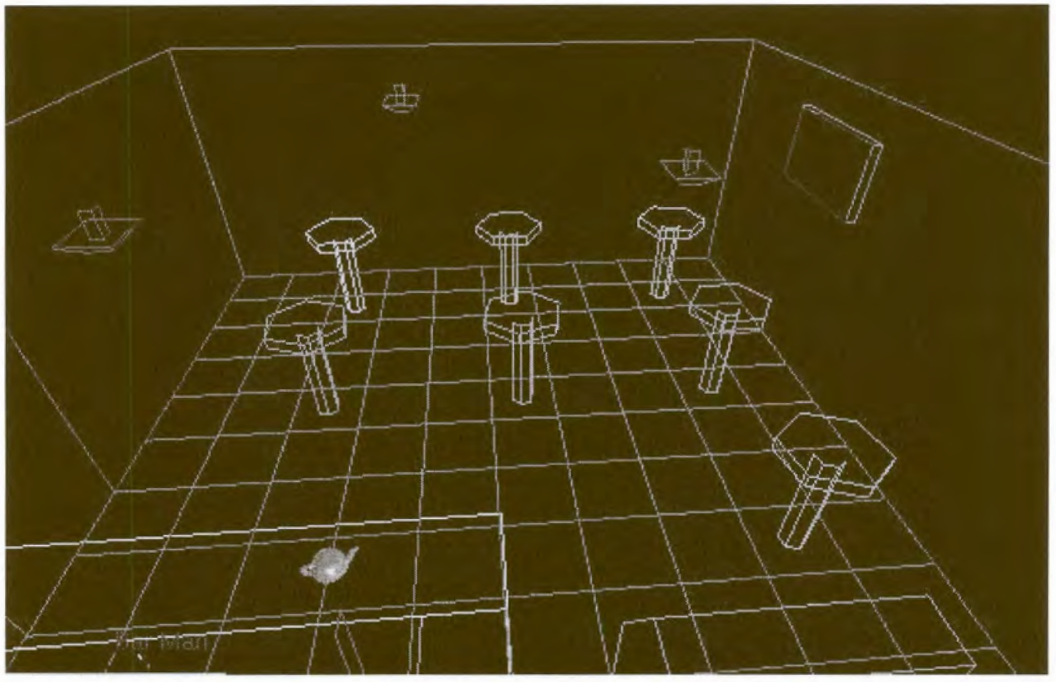

Figure 5.7 The wire-frame view of the virtual pub environment with all the virtual objects.

5.2.2. Texture mapping virtual environment and objects

To make virtual objects more realistic details are added by texture mapping images of real objects or the texture of real object onto the virtual object.

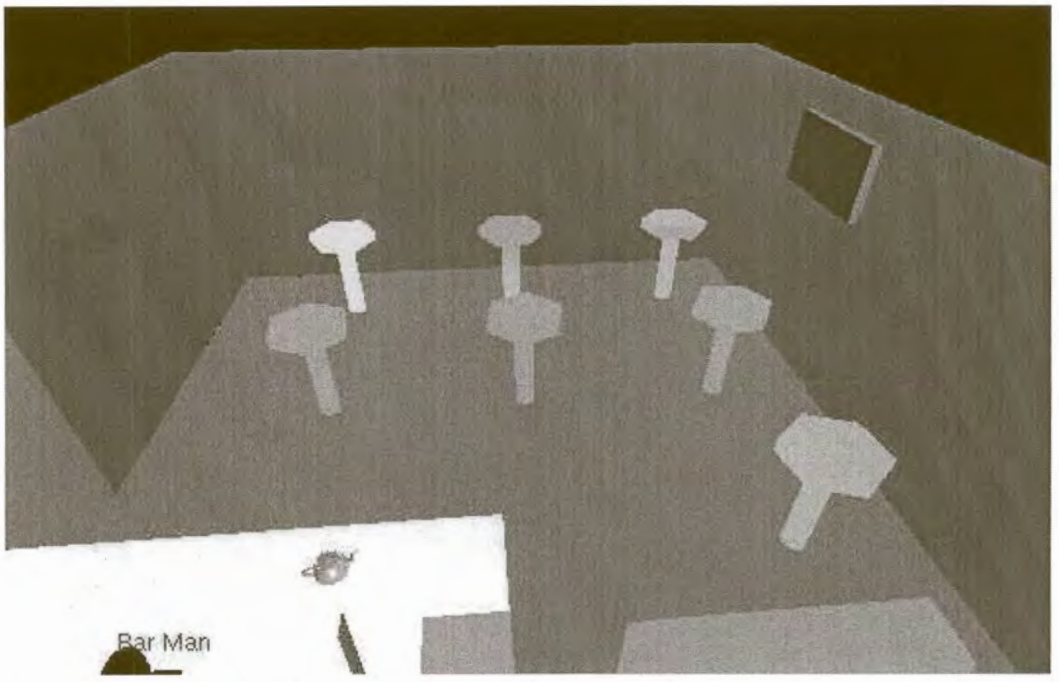

Figure 5.8 The virtual pub environment without textures.

To create a fairly realistic environment a high number of textures needs to be loaded, requiring large amounts of system resources. Therefore, the textures of all the virtual objects and the pub environment are combined into a single high resolution texture and complex texture mapping is applied to all the virtual objects and the virtual environment using this single texture (Figure 5.9). The combined texture is divided into ratio portion, and the texture occupied a specific portion depending on the size of the corresponding virtual object. The reason for this type of texture allocation is that the user will noticed the blur effects and stretching pixels on the object caused by mapping a large virtual object with a small texture. Alternatively, a small texture can be applied to a large surface by tilling the texture. This usually generates a surface with a continuous pattern and the texture is generated seamlessly (without a distinct edge), because the textures can be joined from any direction without any distortion between the repeated textures.

Figure 5.9 A single texture containing all the textures of the virtual pub environment and the virtual objects.

Figure 5.10 shows the result of texture mapping all the virtual objects and the pub environment. After the virtual pub environment is completed, functionality, control and logic are added to the system enabling the user to manipulate the application of the avatars in this virtual pub environment.

Figure 5.10 The result of the virtual pub environment and objects texture mapped with the single combined texture.

5.3. Avatar creation tool

The avatar creation tool is a modification of the expression tool, that allows the user to texture map a face image onto the avatar, customise the body colour and initial

expression of the avatar, instead of texture mapping only the face mesh. The avatar creation tool allows the user to create and design the avatar before it is introduced into the virtual pub environment. The texture mapping process is interactive. The user sees the adjustment made on the face texture and fine tunes the texture mapping of the avatar's face. The user may preview the face expression animations of the avatar in the avatar creation tool. When the user completes the customised design features of the avatar, the avatar together with all its details is sent to the virtual pub environment, without the need to close the avatar creation tool.

The avatar creation tool is interactive (Figure 5.11), and the interaction sequences are simple with enough options and choices for the user to generate an avatar for the virtual pub environment. The interaction devices for the avatar creation tool are the keyboard and the mouse. The basic interaction and selecting options in the avatar creation tool is performed using the mouse, while detail texture mapping adjustment is performed using the keyboard.

Figure 5.11 The avatar creation tool.

The graphical user interface (GUI) of the avatar creation tool and the detail functions of all controls in the GUI are summarised in the Appendix.

5.4. Social Interaction application

The theoretical design of the virtual pub environment and the geometric models of the virtual environment are implemented into a social interaction application. The user interacted with the avatar, after the avatar was sent into the virtual pub environment. The social interaction tool is initialised to include the virtual pub environment and a single avatar playing the role of the barman. Other avatars are first created by the creation tool and are then inserted in the social interaction application.

The interaction between the user and the application is based on selection. First the user selects the avatar and then the task or action that the avatar should perform (Figure 5.12). The avatars' action choices are the actions determined in the previous chapter; namely, drinking, talking to other avatars, watching television and listen to music, while the tasks are 3D interaction tasks which involves rotating the avatar and position it in the virtual pub environment. The interactions between the avatars in the virtual pub are automatic. Once the user selects an action for one of the avatar in the virtual pub environment, the avatar will engage into interaction with another avatar. Although the interaction between the avatars is performed automatically, it requires the user to initiate the action in one of the avatar.

Figure 5.12 The social interaction application with one of the avatar selected for interaction, this was indicated by the bounding box.

The actions of avatars are presented as buttons situated at the top panel of the Social interaction GUI and the 3D interaction tasks are situated at the bottom panel in the Social interaction GUI (Figure 5.13). The mouse is used as the input device for the social interaction application.

Figure 5.13 The actions for the avatars were situated at the top panel in the GUI and the interaction tasks are situated at the bottom panel in the GUI.

The feedback from the avatar is based on visual output onto the screen; the user observes the avatar's emotional expression and body animation when it performs an action or interacts with other avatars in the virtual pub environment. The maximum number of avatars in the virtual pub environment is set to ten, because each avatar had a set of face textures to be loaded and each avatar's action increased the computation of the application. If the maximum number of avatars is not defined in the application, the user might overloaded the virtual pub environment with avatars, which caused the application to performed poorly or even failed to performed execution. The

performance of the application also depends on the number of avatars in the virtual environment and the system resources available for the social interaction application.

The detail functions of all controls in the social interaction GUI will be further discussed in the Appendix.

5.5. Summary

In this chapter, the avatars were generated according to the prototyped models and animated based on the body motion cues defined during the theoretical design of the avatar. Therefore, modelling transformations are applied to the avatar models in OpenGL to achieve correct modelling and motion of body parts.

The theoretical designs of the virtual environment are further implemented into the actual application. The floor of the virtual environment is divided into a grid, for collusion detection and position determination.

The expressive texture tool is modified into the avatar creation tool, instead of only texture mapping a face image onto the face mesh, the dress colour of the avatar can be selected by the user and previewed. The user can create the avatar in the avatar creation tool and then send the data of the avatar into the virtual pub environment.
The social interaction application described in this chapter is one possible application of the avatar and provides basic interaction between different avatars in the virtual pub environment, as well as the user-avatar interaction. The interactions in the social interaction application have a simple to use graphical interface. The selected avatar performs actions selected by the user and interacts automatically with other avatars if the action performed affects other avatars in the virtual environment.

The next chapter concludes the findings and results of this thesis, discusses future work and possible improvements to the development of avatars and social interactive application.