Computer Graphics
&
Visualization Project

Cross-Platform 3D Chess

by:
ComputeristGeek

The Files Used

1. **Texture.cpp** includes: **iostream**,**GL/glut.h**
along with
    1. **Bwood.bmp**
    2. **BwoodChoice.bmp**
    3. **BWTop.bmp**
    4. **wood.bmp**
    5. **woodBottom.bmp**
    6. **Wwood.bmp**
    7. **WwoodChoice.bmp**

Texture.cpp and the .bmp files are the texture loader and the image textures respectively

2. **Chars.cpp** includes:
“**Texture.cpp**”,**cstdio**,**iostream**,**GL/glut.h**
This file contains the graphics core of the chess game including the shapes and structures of all objects.
It also links between graphics and bitboard.

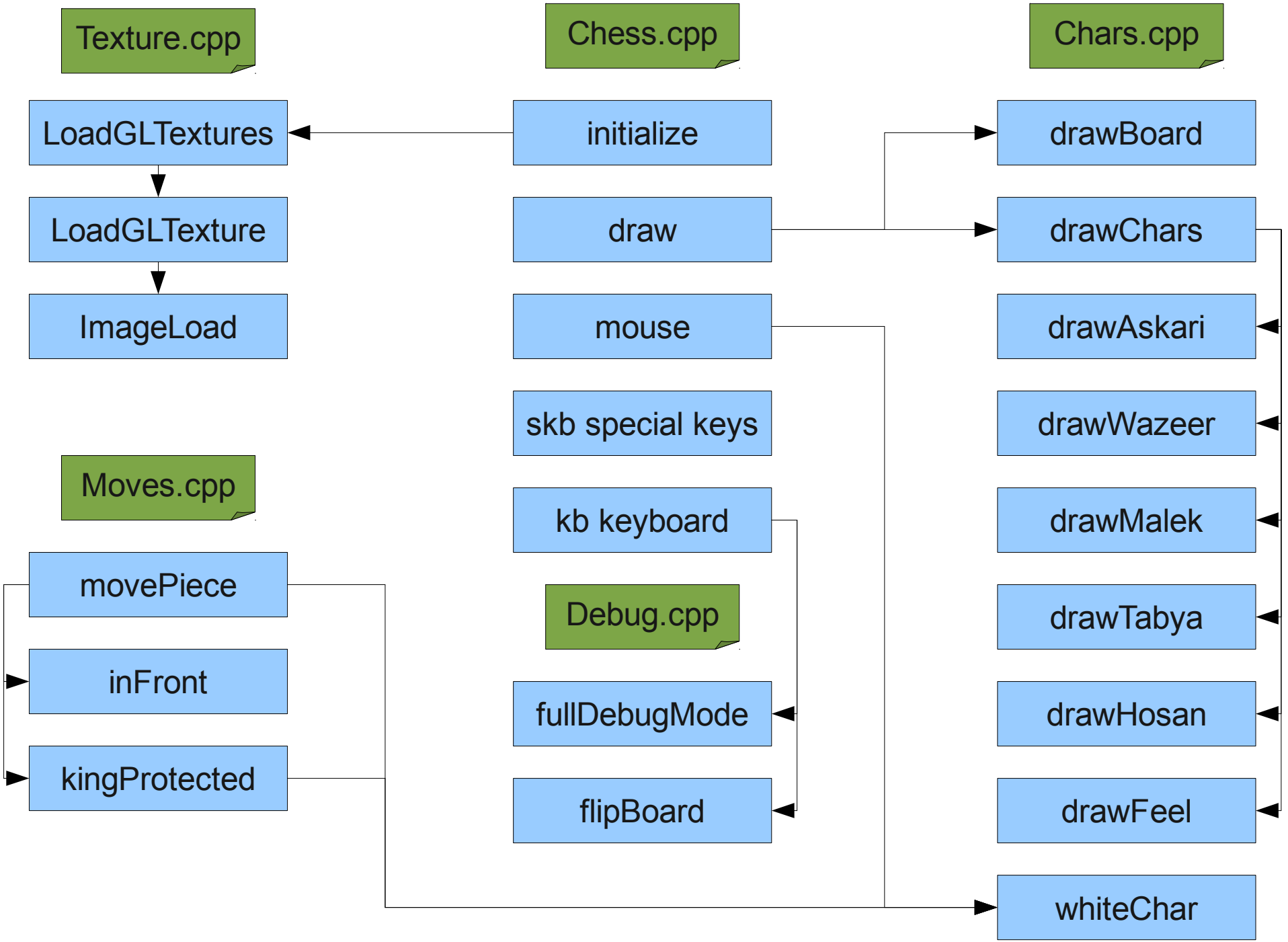
3. **Moves.cpp** includes: “**Chars.cpp**”,
“**Debug.cpp**”,**cmath**
This file contains the engine/rules core of the chess game including the basic moving rules, the basic killing/conquering rules and checking for the king's protection.

4. **Chess.cpp** includes:
"Moves.cpp" ,**cstdio**,**iostream**,**GL/glut.h**
This file contains the main drawing, mouse, keyboard, special keys functions.

5. **Debug.cpp** includes: **cstring**
This file contains a number of functions for debugging through cheats.

The Functions Used

A simple chart is made to show the functions in the next page.


Texture.cpp
LoadGLTextures
LoadGLTexture
ImageLoad
Chess.cpp
initialize
draw
mouse
skb special keys
kb keyboard
Debug.cpp
fullDebugMode
flipBoard
Chars.cpp
drawBoard
drawChars
drawAskari
drawWazeer
drawMalek
drawTabya
drawHosan
drawFeel
whiteChar
Moves.cpp
movePiece
inFront
kingProtected

- The function **ImageLoad**, loads the image address sent to it into memory.
- The function **LoadGLTexture** calls the **ImageLoad** function and links the image with the texture number sent to it.
- The function **LoadGLTextures** calls the **LoadGLTexture** function with the specified images and their respective texture number to link.

The LoadGLTextures function is only called once during initialization in the **initialize** function in Chars.cpp

- The function **initialize** in Chess.cpp sets the options for display and sets the selecting buffer.
- The function **draw** is the default drawing function that is used for both, rendering and selection, it calls the **drawBoard** and **drawChars** functions.
- The function **mouse** is the default mouse function that is used for rotation and/or selection, it calls functions **whiteChar** and **movePiece** to check for turn/color of piece and whether the move is allowed or not.
- The function **skb** is the default special-keys function that is used to rotate the board and pieces.
- The function **kb** is the default keyboard function which is used for entering cheats for debugging calling **fullDebugMode** and/or **flipBoard** functions.

- The function **drawBoard** in Chars.cpp is used to either draw the chess board in rendering mode, or draw a 2D square as a board for selection mode (for more speed and accuracy since only the black/white part is needed during selection)
- The function **drawChars** checks the board for the distribution of the pieces and calls the respective character in its place.
- The functions **drawAskari**, **drawTabya**, **drawHosan**, **drawFeel**, **drawWazeer**, **drawMalek** are all used to draw the respective characters (Askari/Pawn, Tabya/Rook, Hosan/Knight, Feel/Bishop, Wazeer/Queen, Malek/King) in their respective places.
- The function **whiteChar** returns true if the character number sent is of a white character and false if it's of a black one.

- The function **movePiece** in Moves.cpp checks whether the specified move for the specified character is allowed or not, it also calls functions **inFront** and **KingProtected**.
- The function **inFront** checks whether there is any pieces in between the character and its destination because some characters can not bypass others that are in their way.
- The function **KingProtected** simulates the move and checks whether the King will be protected after it or not.

The Variables Used

- unsigned int PickBuffer[PICK_BUFFER_SIZE]

Used in Chess.cpp globally as the array that stores the information returned from graphics card about the object(s) picked.

- bool firstTime=true;

Used in Chess.cpp globally to know whether it was the first time to display or not, useful for making an appropriate rotation where all characters are visible and to tell function drawChars to initialize their positions as the beginning of the game.

- int chosen=-1;

Used in Chess.cpp globally to store the number of the object chosen, it is initialized as -1 because no object holds the number -1 and the graphics card by default returns -1 if nothing was picked.

- float Xmouse=0, Ymouse=0

Used in Chess.cpp globally to globally store the position x,y where the mouse button was clicked for selection, to be used by the draw function in selection mode.

- char message[]={'\0','\0','\0','\0','\0','\0','\0','\0'}

Used in Chess.cpp globally to store the characters entered through keyboard for the debug/cheat

mode.

- float width=glutGet(GLUT_WINDOW_WIDTH) , height=glutGet(GLUT_WINDOW_HEIGHT);

Used in Chess.cpp in draw function to get the actual width and height of the window.

- int viewport[4];

Used in Chess.cpp in draw function to store the current viewport/projection.

- int angle;

Used in Chess.cpp in mouse function to store the angle (-ve rotation or +ve rotation)

- int prevChosen=chosen;

Used in Chess.cpp in mouse function to store the previously chosen character for usage later.

- int Nhits=glRenderMode(GL_RENDER);

Used in Chess.cpp in mouse function to store the number of hits (ray intersections with each object) resulting from selection.

- int nitems=PickBuffer[index++];

Used in Chess.cpp in mouse function to pick the number of items for each hit (each object/hit has an array of items for it)

- int Board[8][8]

Used in Chars.cpp globally to simulate the chessboard.

- int RenderMode

Used in Chars.cpp globally to store the RenderMode that we are in (GL_SELECT or GL_RENDER) to avoid calling the function to check.

- int charsPosition[32];

Used in Chars.cpp globally to store the position of each character (opposite of Board[8][8]) for quicker retrieval of character positions when needed.

- bool flipped=false;

Used in Chars.cpp globally to store whether the board is flipped or not.

- int num=8*i+j+32;

Used in Chars.cpp in drawBoard function to convert from 2D array representation to 1D.

- GLUquadricObj

Used in all drawing functions except drawBoard for drawing quadratic objects like spheres/cones etc. to make up the end shapes for the characters.

- float XX=-1+0.125,YY=0.25-DIV16,ZZ=-1+0.125;

Used in Chars.cpp in each drawing function (except drawBoard) to transform it to the proper position needed holding the top left end of the board's position and translating from it.

- bool turn=true;

Used in Moves.cpp globally to store the turn (white's turn=true, black's turn=false).

- int ifrom=from/8,jfrom=from-8*ifrom;
- int ito=to/8,jto=to-8*ito;

Used in Moves.cpp in KingProtected and movePiece to get the I and j coordinates from 1D array.

- int simulateBoard[8][8],simcharsPosition[32];

Used in Moves.cpp in KingProtected to simulate the chessboard and the charsPosition arrays to freely change positions then check if the king is endangered without actually modifying the real arrays.

- bool white=true;

Used in Moves.cpp in KingProtected to store whether it's the white or black move.

- int test=simulateBoard[ito][jto];

Temporary variable.

- int c0=(white?0:-8),c1=(white?0:-1),c2=(white?0:-2);

Used in Moves.cpp in KingProtected to convert constants from blackPieces to whitePieces, c0 for Askari, c1 and c2 for types that have 1 piece or 2 pieces, respectively.

- int Askari0=blackAskari0+c0,Askari7=blackAskari7+c0
- int Wazeer=blackWazeer+c1,Malek=blackMalek+c1;
- int Feel0=blackFeel0+c2,Feel1=blackFeel1+c2,Hosan0=blackHosan0+c2,Hosan1=blackHosan1+c2,Tabya0=blackTabya0+c2,Tabya1=blackTabya1+c2;
- int Mali=simcharsPosition[white?whiteMalek:blackMalek]/8,Malj=simcharsPosition[white?whiteMalek:blackMalek]-Mali*8;

Used in Moves.cpp in KingProtected to convert white pieces to black pieces through the constants.

- int tempi=Mali+inci,tempj=Malj+incj;
- int otherMali = simcharsPosition[Malek]/8,otherMalj = simcharsPosition[Malek]-otherMali*8;

Used in Moves.cpp in KingProtected to determine the position of the other Malek/King

- int begin=(white?1:6);

  Used in Moves.cpp in KingProtected and movePiece to determine which side did this object start from (up or down).

- bool testing[]={true,true,true,true,true,true,true,true};

Used in Moves.cpp in KingProtected used to store whether to continue checking in this side of the king or not (left, right, bottom, up and diagonal checks).

- int inc=(i1<i2)?j1:j2;

Used in Moves.cpp in inFront to determine how many increments should be applied depending on the I chosen.

- bool onlyOneMoveForward=(begin==1)?(ito-ifrom==1):(ito-ifrom==-1);
- bool twoMovesForward=(begin==1)?(ito-ifrom==2):(ito-ifrom==-2);

Used in Moves.cpp in movePiece to know how many moves are allowed (special rule for

askari/pawn).

- GLuint texture[max];

Used in Texture.cpp globally to store the number of textures to be used, one integer for each texture.

- struct Image {
- unsigned long sizeX;
- unsigned long sizeY;
- char *data;
- };

Used in Texture.cpp globally to store the Image as size and data, width and height and the data.

- FILE *file;

Used in Texture.cpp in ImageLoad to open the file as a binary file to read the image.

- unsigned long size;

Used in Texture.cpp in ImageLoad to store size of the image in bytes.

- unsigned short int planes;

Used in Texture.cpp in ImageLoad to store number of planes in image (must be 1).

- unsigned short int bpp;

Used in Texture.cpp in ImageLoad to store number of bits per pixel (must be 24) .

- char temp;

Used in Texture.cpp in ImageLoad to convert bgr to rgb color.

- Image *image1;

Used in Texture.cpp in ImageLoad to store the image loaded to bind it to the texture.

- bool Debug=false,freee=false;

Used in Debug.cpp for debugging purposes whether to allow debugging statements to be issued or to allow free movement of the pieces without constraints.

Please refer to the presentation for more information.