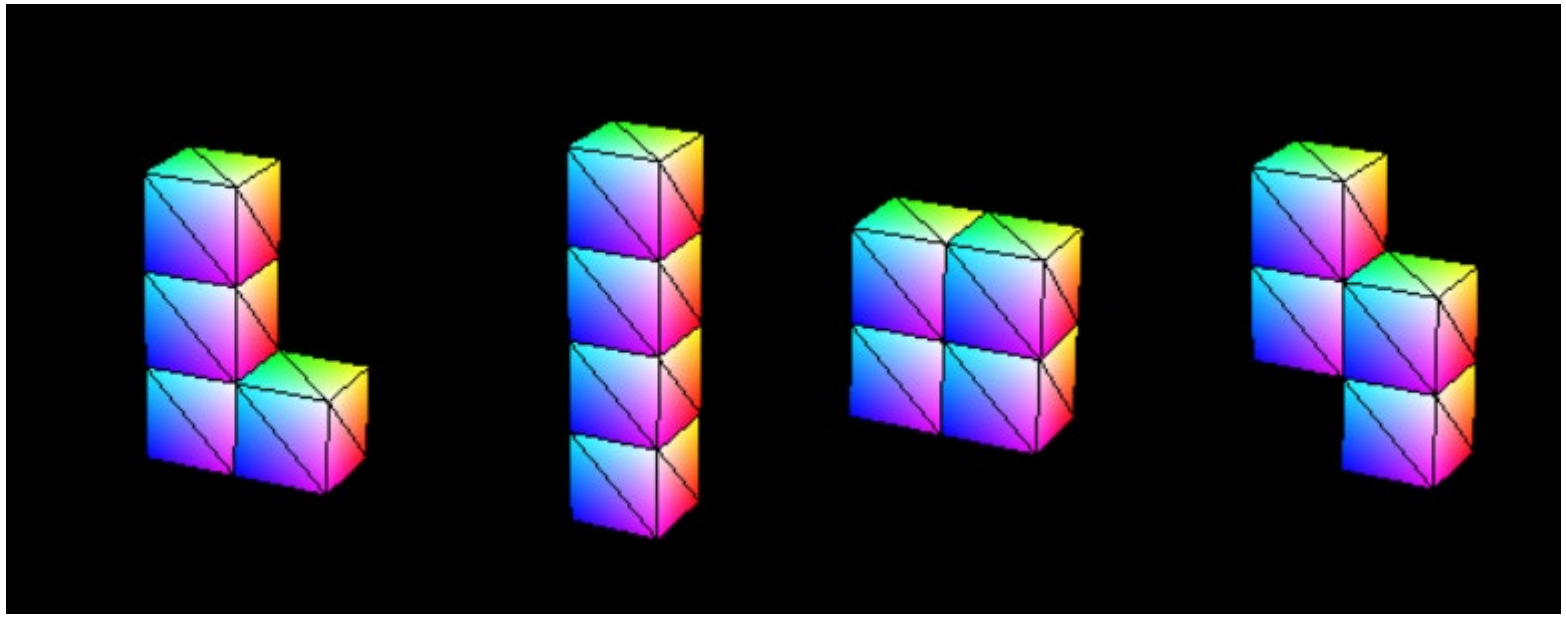
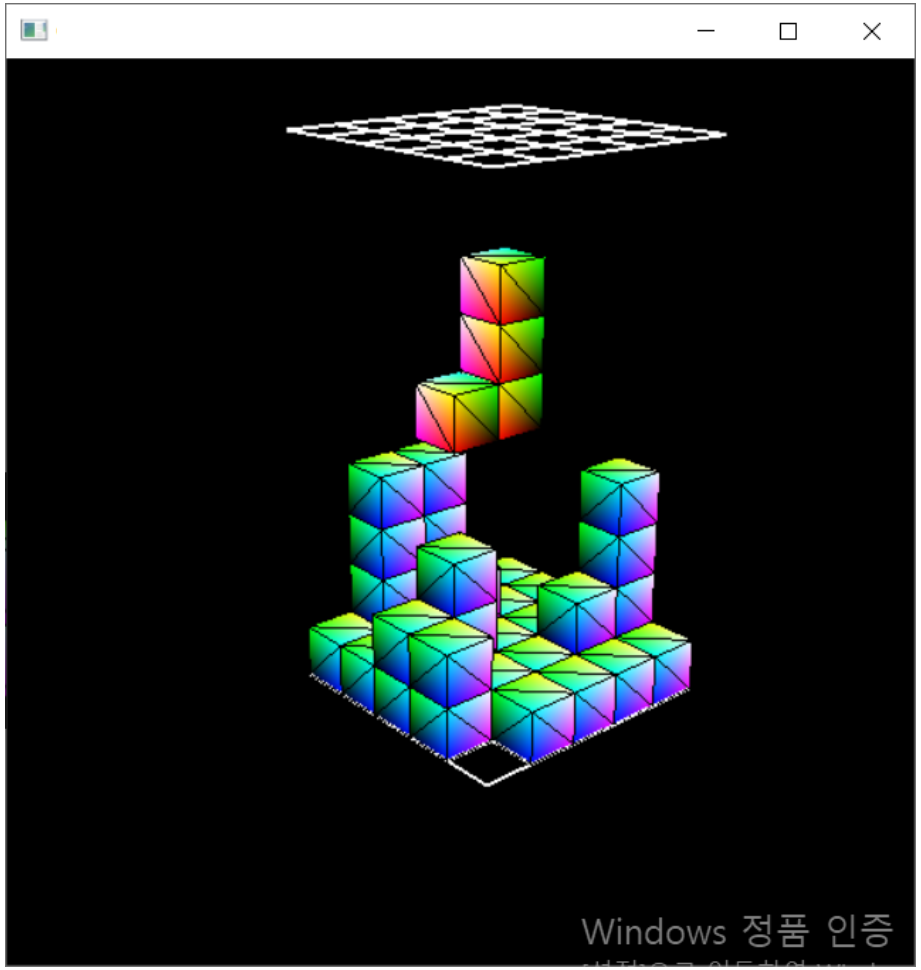
3D Tetris

- Goal
  - To make a 3D Tetris game that supports the following four shapes of blocks:



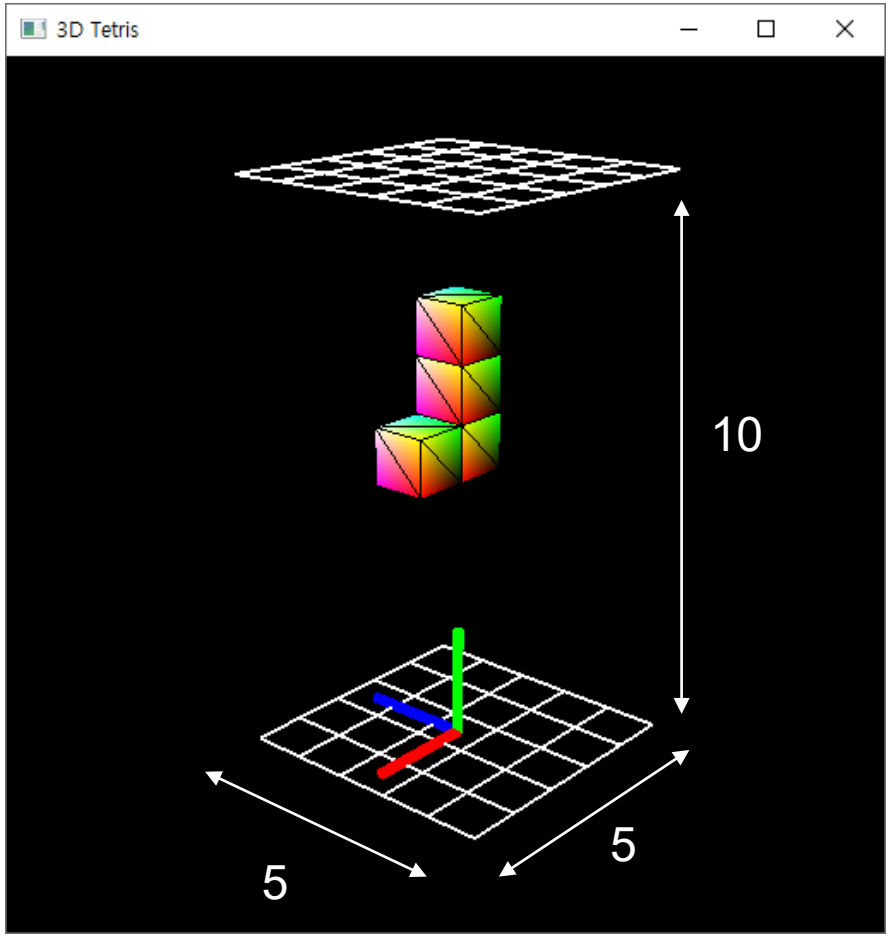
L shape I shape Box shape N shape



Requirement 1

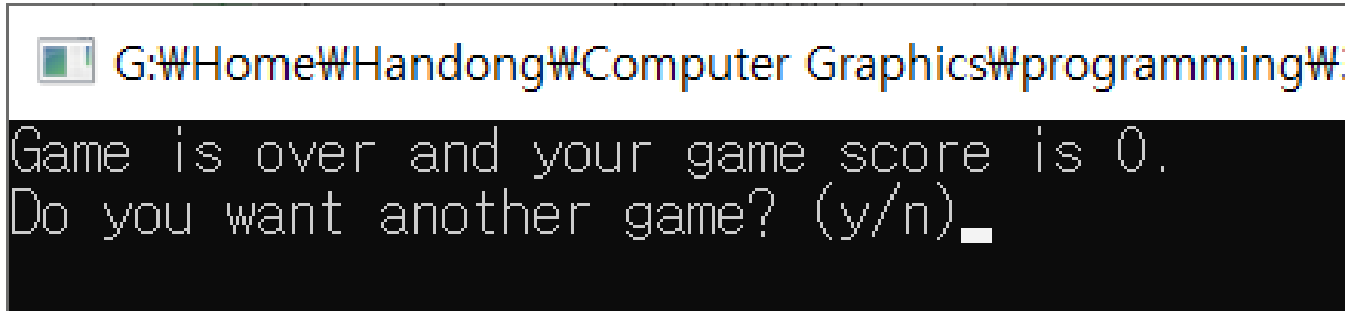
- The size for the stack of blocks should be greater than or equal to **5 x 10 x 5**.
- A block **keeps going down** by one cell every one second. You may utilize the following code for this:

```
#include <time.h>

void idle()
{
    static clock_t prev_time = clock();
    clock_t curr_time = clock();
    if (1.0 * (curr_time - prev_time) / CLOCKS_PER_SEC > 1.0) {
        ... Update the block to make it go down. ...
        glutPostRedisplay();
    }
    prev_time = curr_time;
}

void main(int argc, char** argv) {
    ...
    glutIdleFunc(idle);
    ...
}
```

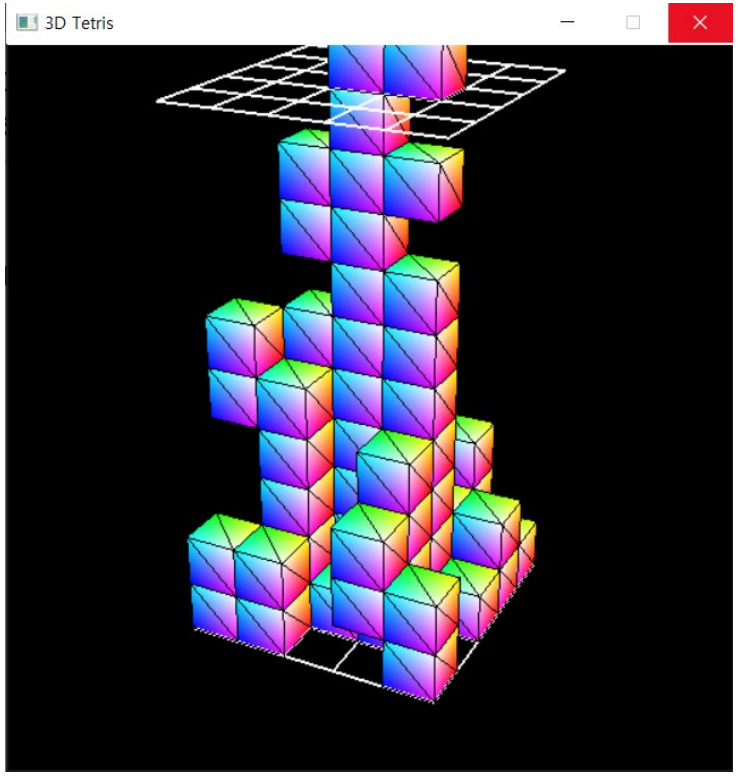


- If the moving block **touches the ground or any other existing block below it**, make it stacked over the ground or existing block. Then, another new block, randomly chosen, should come down from the top again.

- If the current height of the stack of blocks is greater than the maximum height, **the game is over**, and the program should ask the user to do another round.



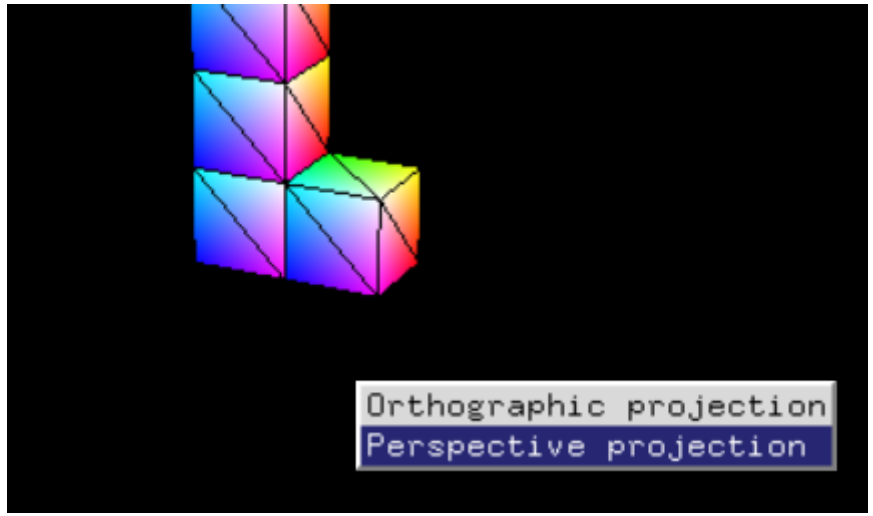
- The **game score** is the total number of layers that have been cleared from the stack.
- As soon as any layer of the stack is **completely filled with blocks**, it should be cleared.



Requirement 2

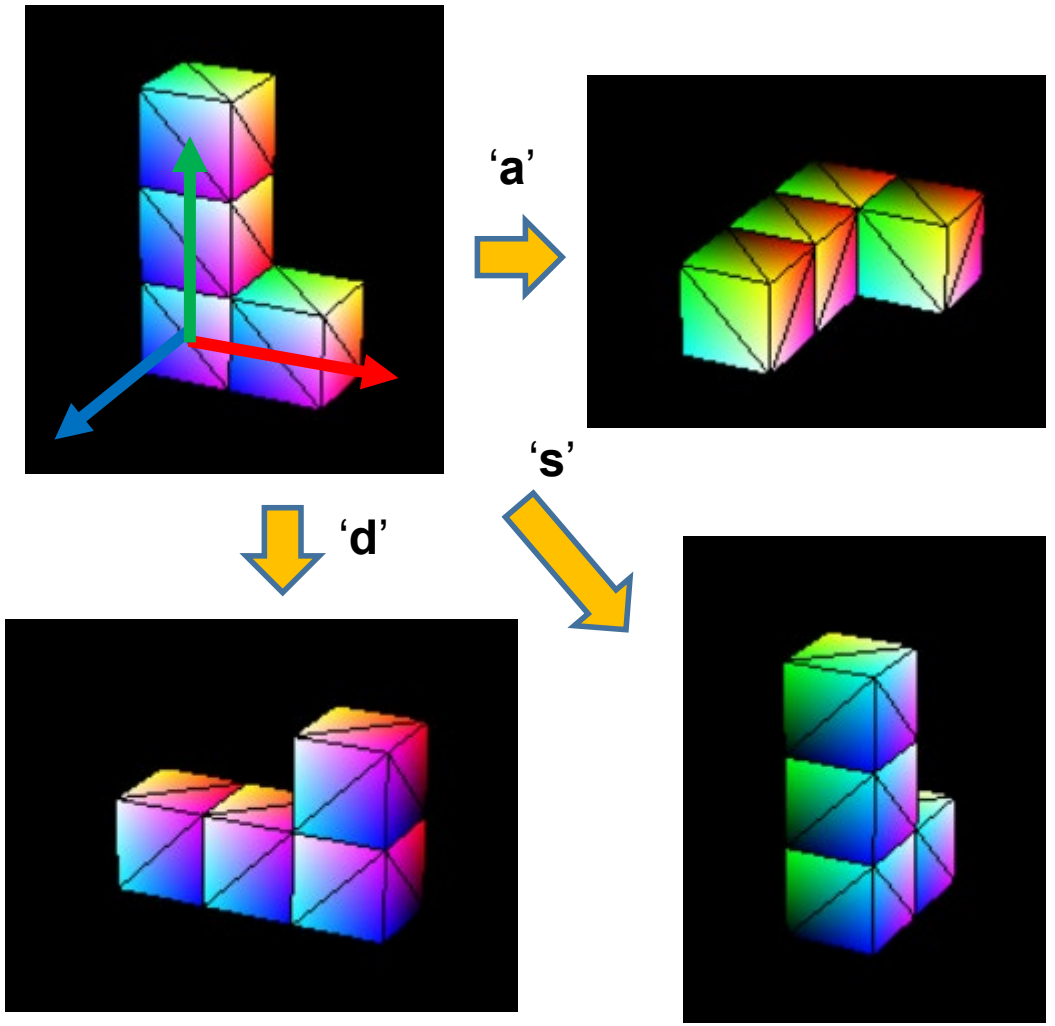
- The interactive viewing control of HW9 must be supported:
  (a) **Mouse Left Button + Alt + Drag**: Works as the **Tumble tool** does
  (b) **Mouse Middle Button + Alt + Drag**: Works as the **Track tool** does
  (c) **Scroll up / down**: Works as the **Zoom tool** does
  (d) **Scroll up / down + Alt**: Works as the **Dolly tool** does

- When the user clicks the right mouse button, the application should show a pop-up menu to let the user select one of the orthographic and the perspective projection modes.

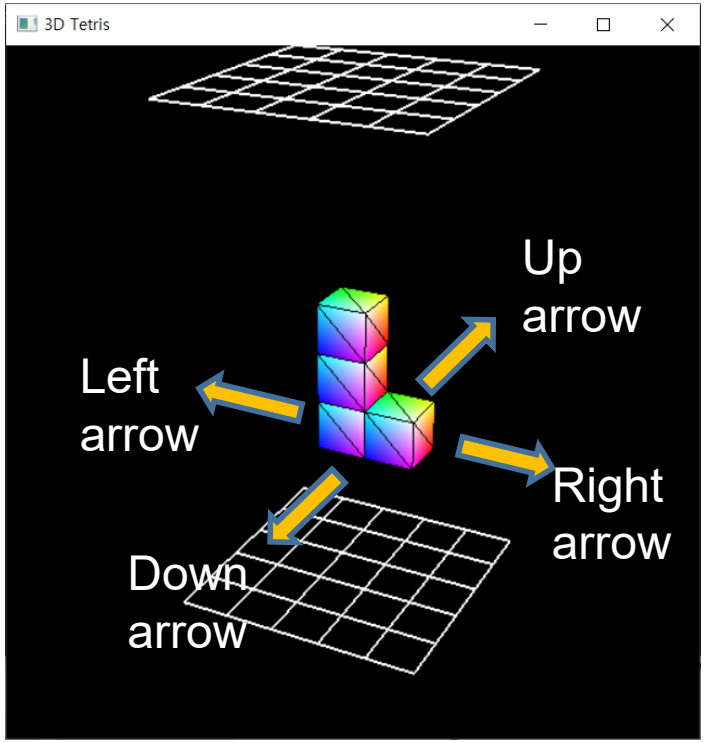


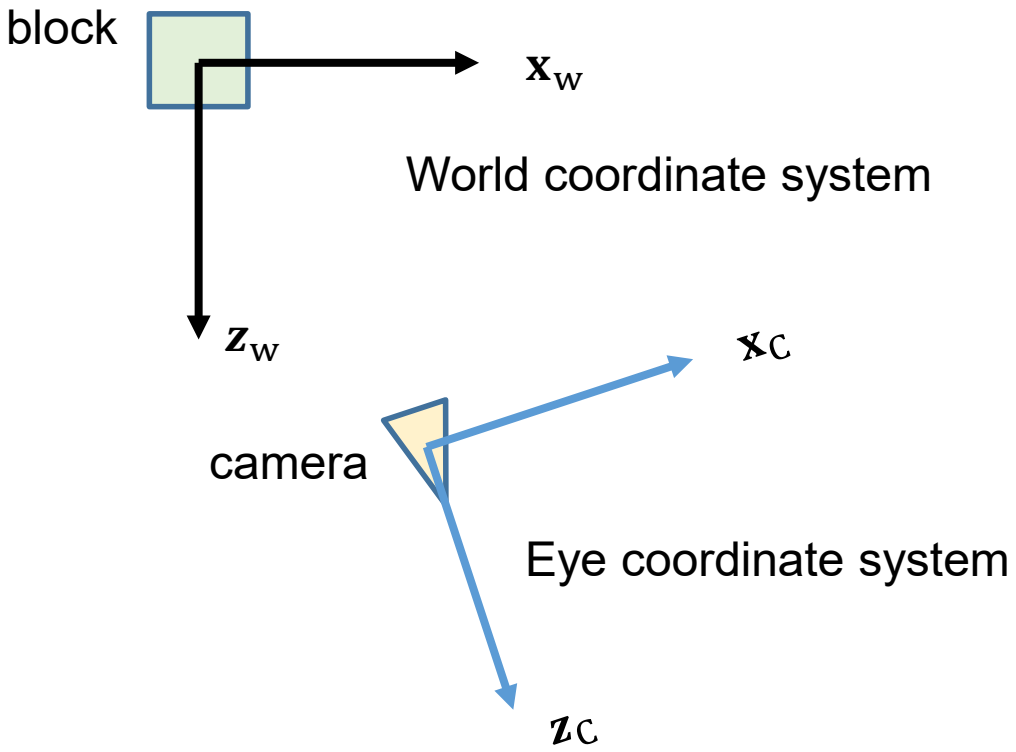
Requirement 3: Block rotation and position control

- '**a**' key
  - Rotate a block about the x-axis by 90 degrees.
- '**s**' key
  - Rotate a block about the y-axis by 90 degrees.
- '**d**' key
  - Rotate a block about the z-axis by 90 degrees.
- If the rotated block exceeds the empty stack space or penetrates the existing blocks, that rotation should not be allowed.



- **Left Arrow / Right Arrow**
  - Move a block to the left / to the right on the x-z plane with respect to the camera view.
- **Up Arrow / Down Arrow**
  - Move a block up / down on the x-z plane with respect to the camera view.





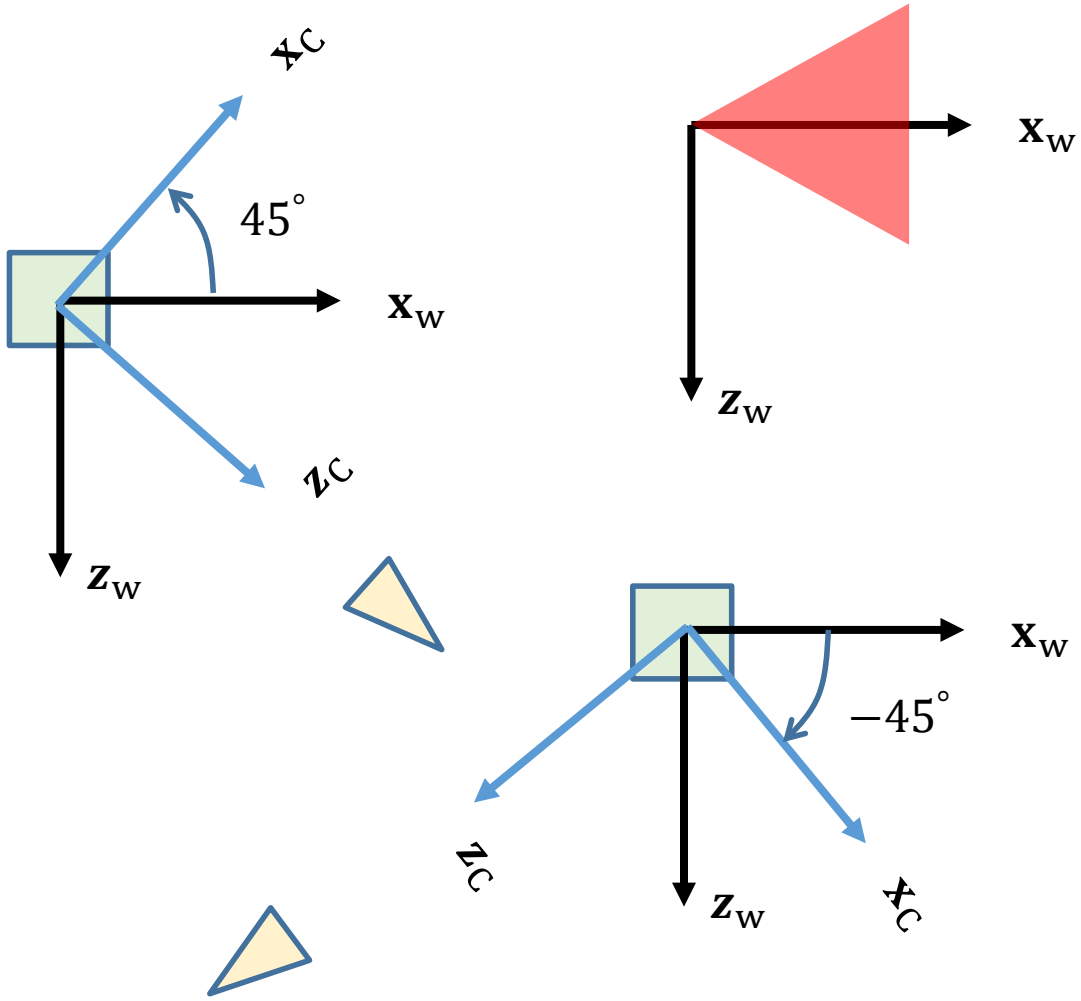
Left Arrow: move the block by -1 cell along x_w

Right Arrow: move the block by +1 cell along x_w

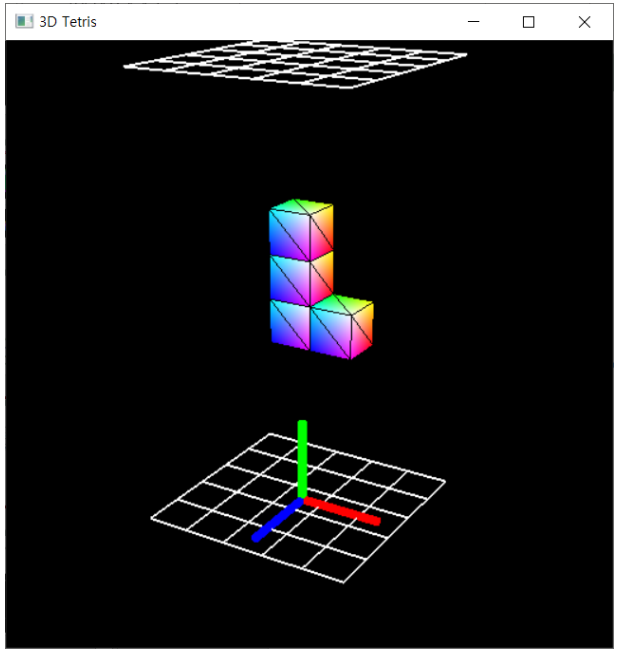
Up Arrow: move the block by -1 cell along z_w

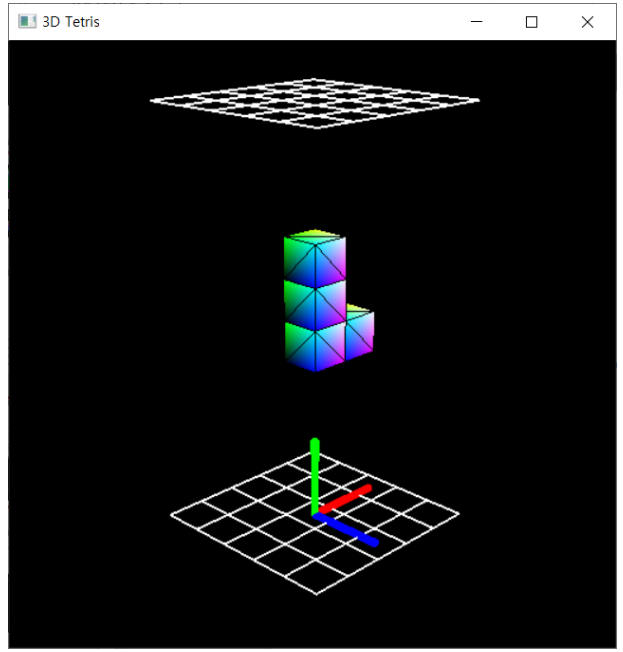
Down Arrow: move the block by +1 cell along z_w

Case 1: -45° < Angle < 45°

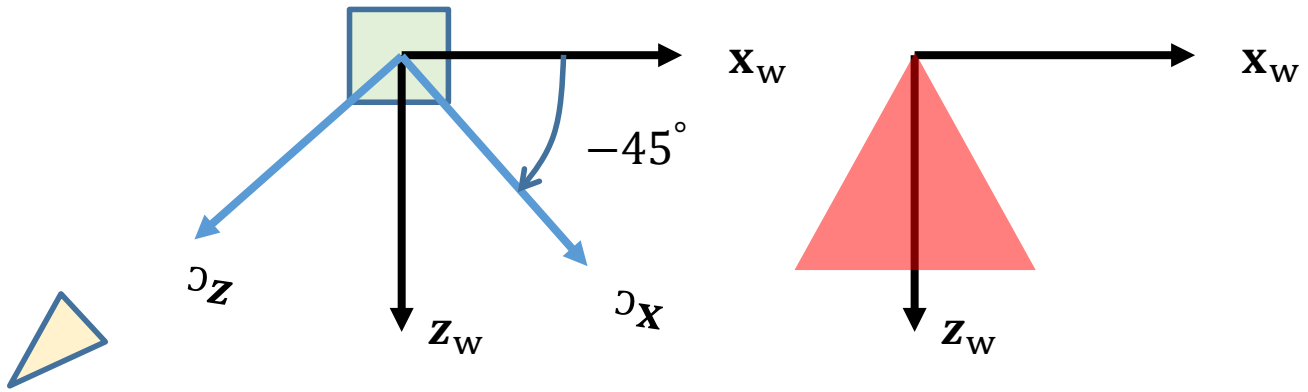


Left Arrow	Move by -1 cell along x_w
Right Arrow	Move by +1 cell along x_w
Up Arrow	Move by -1 cell along z_w
Down Arrow	Move by +1 cell along z_w

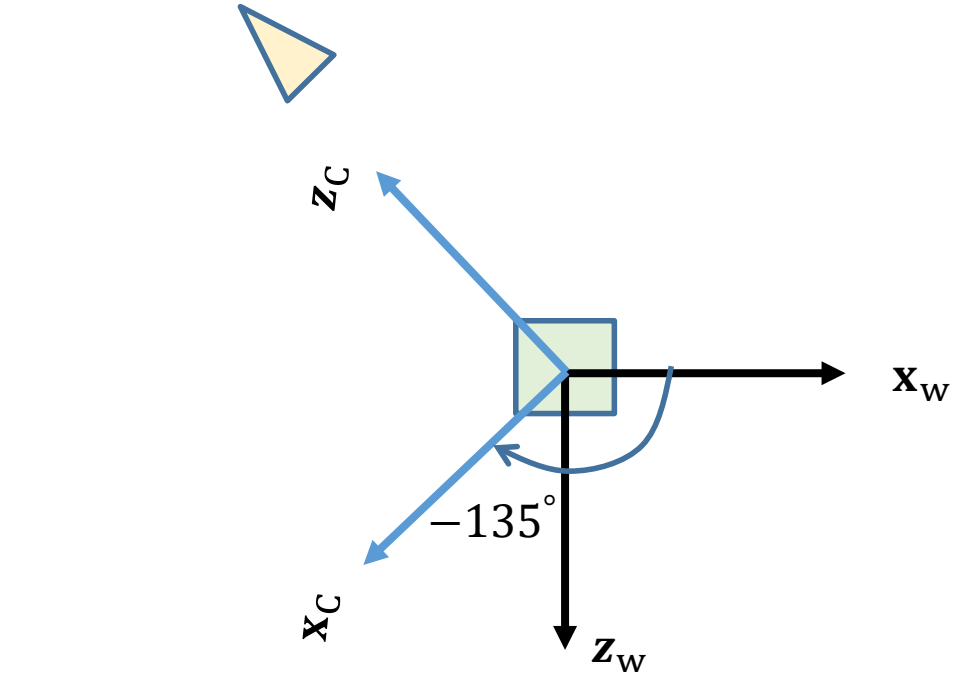


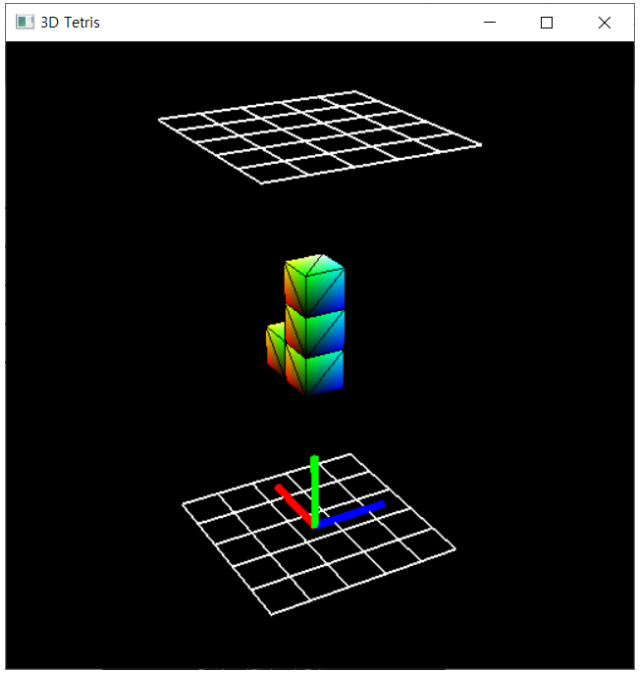


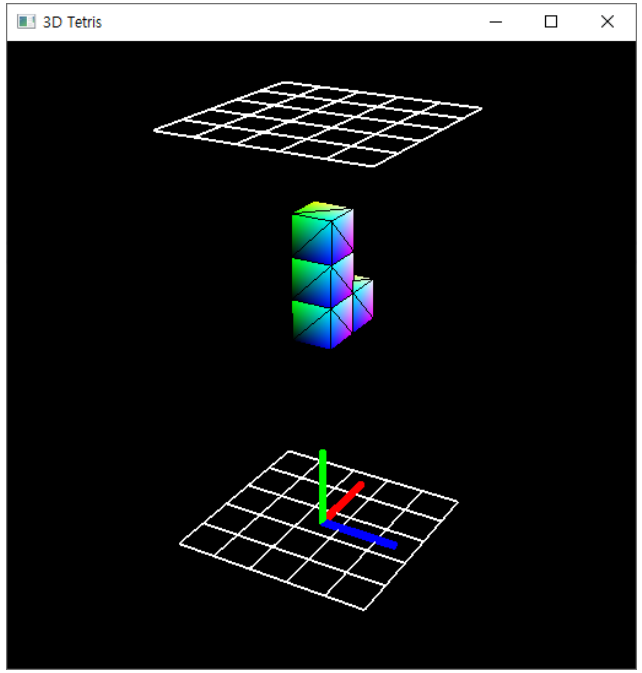
Case 2: $-135^{\circ} < \text{Angle} \leq -45^{\circ}$



Left Arrow	Move by -1 cell along z_w
Right Arrow	Move by +1 cell along z_w
Up Arrow	Move by +1 cell along x_w
Down Arrow	Move by -1 cell along x_w

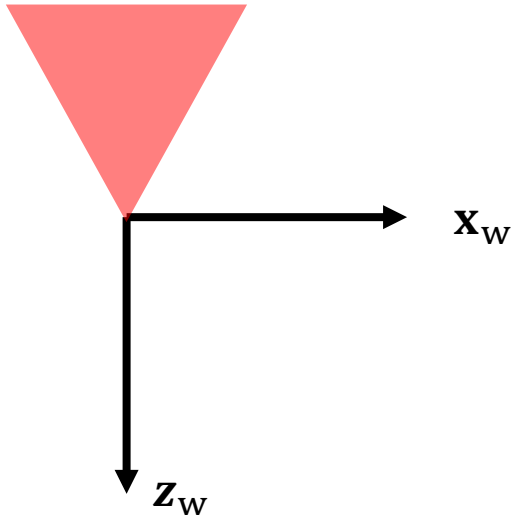






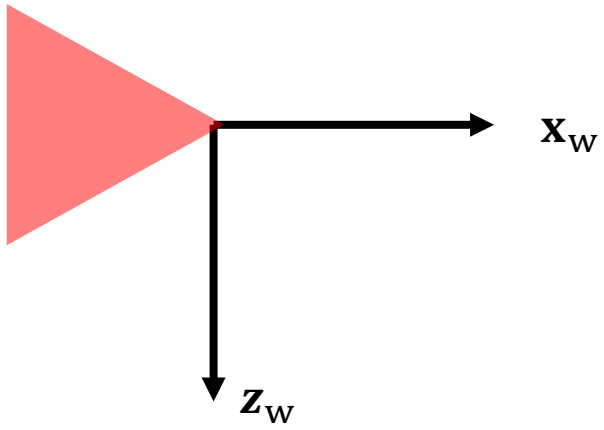
Similarly,

Case 3: $45^\circ \leq$ Angle $< 135^\circ$



Left Arrow	Move by +1 cell along $\mathbf{z}_w$
Right Arrow	Move by -1 cell along $\mathbf{z}_w$
Up Arrow	Move by -1 cell along $\mathbf{x}_w$
Down Arrow	Move by +1 cell along $\mathbf{x}_w$

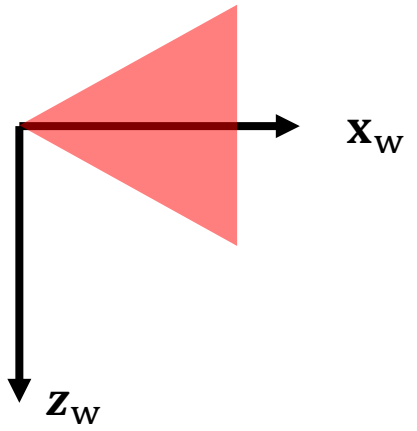
Case 4: otherwise



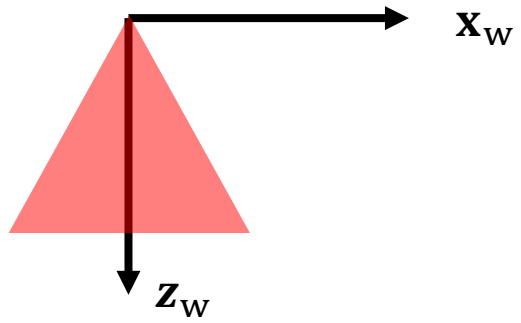
Left Arrow	Move by +1 cell along $\mathbf{x}_w$
Right Arrow	Move by -1 cell along $\mathbf{x}_w$
Up Arrow	Move by +1 cell along $\mathbf{z}_w$
Down Arrow	Move by -1 cell along $\mathbf{z}_w$

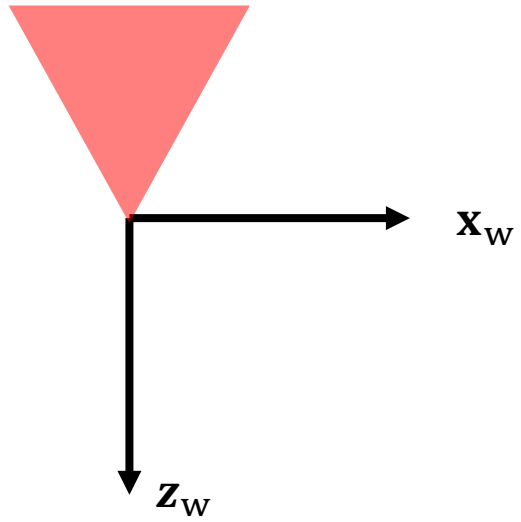
- Example code

```
void specialkey(int key, int x, int y)
{
    using namespace glm;
    mat4 VT = transpose(camera.get_viewing());
    vec3 x_axis(VT[0]);

    x_axis[1] = 0;
    x_axis = glm::normalize(x_axis);
    float angle = (float)(180.0f * acosf(glm::dot(x_axis, vec3(1, 0, 0))) / M_PI);

    if (angle < 45)
    {
        switch (key)
        {
        case GLUT_KEY_LEFT:block.move(-1, 0,  0); break;
        case GLUT_KEY_RIGHT:block.move(+1, 0,  0); break;
        case GLUT_KEY_UP:block.move( 0, 0, -1); break;
        case GLUT_KEY_DOWN:block.move( 0, 0, +1); break;
        }
    }
```

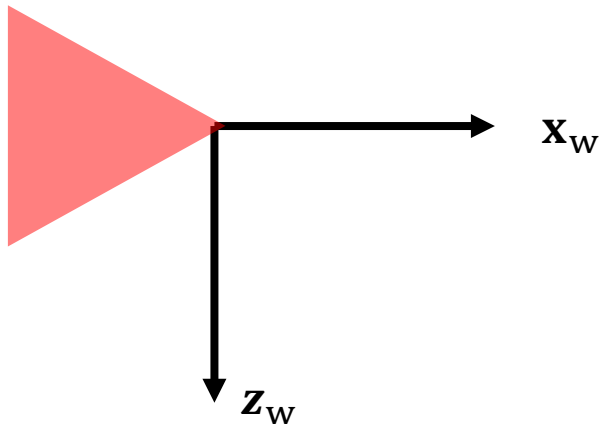


```
else if (angle < 135)
{
    if (x_axis[2] > 0)
    {
        switch (key)
        {
        case GLUT_KEY_LEFT:block.move(0, 0, -1); break;
        case GLUT_KEY_RIGHT:block.move(0, 0, +1); break;
        case GLUT_KEY_UP:block.move(+1, 0, 0); break;
        case GLUT_KEY_DOWN:block.move(-1, 0, 0); break;
        }
    }
    else
    {
        switch (key)
        {
        case GLUT_KEY_LEFT:block.move(0, 0, +1); break;
        case GLUT_KEY_RIGHT:block.move(0, 0, -1); break;
        case GLUT_KEY_UP:block.move(-1, 0, 0); break;
        case GLUT_KEY_DOWN:block.move(+1, 0, 0); break;
        }
    }
}
```





```
    else
    {
        switch (key)
        {
        case GLUT_KEY_LEFT:block.move(+1, 0,  0); break;
        case GLUT_KEY_RIGHT:block.move(-1, 0,  0); break;
        case GLUT_KEY_UP:block.move( 0, 0, +1); break;
        case GLUT_KEY_DOWN:block.move( 0, 0, -1); break;
        }
    }
    ... You may do any other operations here if needed. ...
}
```



- What to submit:
  - A **zip file** that compresses the following files:
    - **Project source files** except libraries.
      - Clean your project before compression by selecting **Build → Clean Solution** in the main menu.
    - **A video file** that shows how the game works.
    - **A short report** (more than one A4 page) that describes
      - What functions were implemented and what functions were missing
      - How to play the game
      - What you learned or discussion
      - Any extra efforts (which may give you extra credits) if there is any
  - File name format
    - **proj_000000.zip**, where 000000 must be replaced by your own student ID.
- Due date: **23:59, June 25 (Friday, Week 17)**