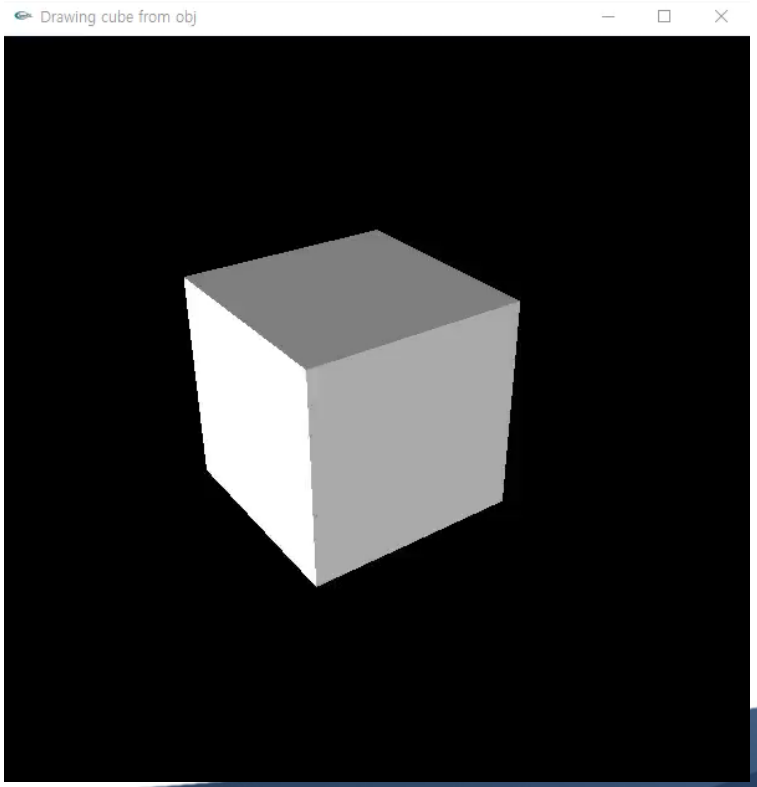
LAB I
Week 06

Seoul National University
Graphics & Media Lab
HyeonSeung Shin

Today's Mission

- Load data from OBJ file (cube)
  - Use STL vector
  - Define vertex, face, obj class
- Access member variables only through member functions
- Rotate the cube using idle function



Standard Template Library (STL)

- It brings huge innovation in C++ programming.

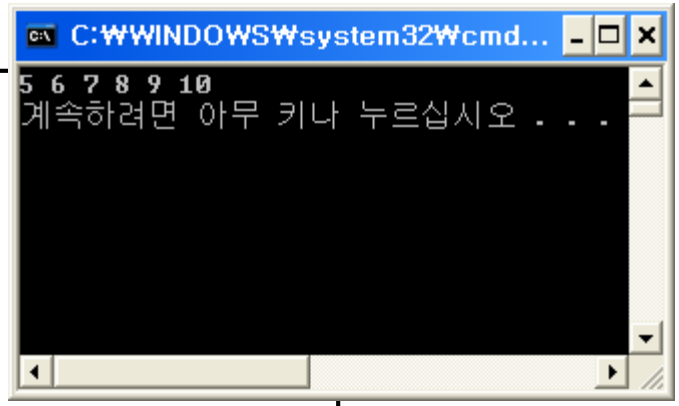
- It is **generic library** (i.e., based on **template**) which provides
  - Container (ex. vector, string)
  - Iterators
  - Algorithms
  - Functors

- It is very **general, efficient, and easy to use**.

Standard Template Library (STL)

- `vector<T>` Library
  - dynamic array
    - similar to array, but size is flexible
    - http://en.wikipedia.org/wiki/Dynamic_array
  - takes care of managing the memory associated with storing the elements
  - implemented as a class template

```
#include <iostream>
#include <vector>

void main() {
   std::vector<int> ivec;
   for(int i=5;i<=10;++i)
      ivec.push_back(i);

   for(std::vector<int>::size_type i=0;i!=ivec.size();++i)
      std::cout << ivec[i] << " ";
   std::cout << std::endl;
}
```



vector Library

- `vector<T>` Library
  - dynamic array
    - similar to array, but size is flexible
    - http://en.wikipedia.org/wiki/Dynamic_array
  - http://www.cplusplus.com/reference/stl/vector/
  - http://en.wikipedia.org/wiki/Vector_(C%2B%2B)

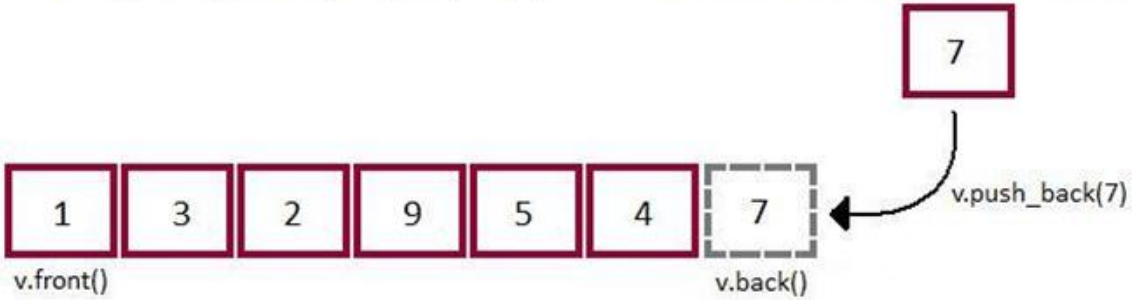
  - Member functions
    - size : Return size
    - resize : Change size
    - push_back : Add element at the end
    - pop_back : Delete last element
    - begin : Return iterator to beginning
    - end : Return iterator to end
    - ……

vector Library

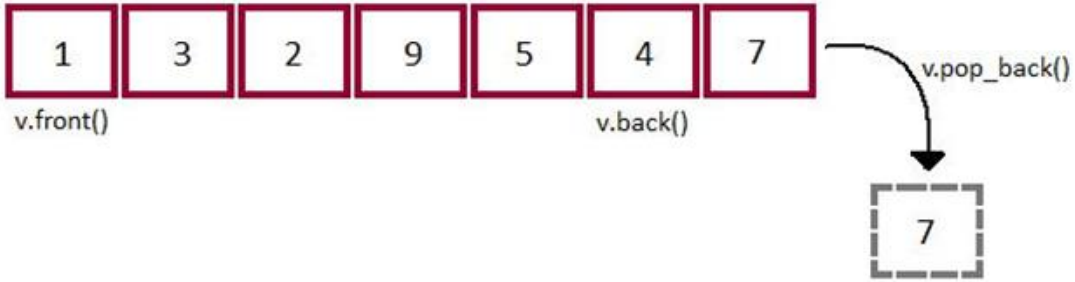
- `vector<T>` Library

Vector visualization

Semantically picturing the vector *push_back(data)* operation. Note the size of the vector increases from 6 to 7 after the operation.



Semantically picturing the vector *pop_back()* operation. Note the size of the vector decreases from 7 to 6 after the operation.



Semantically picturing the vector *resize(int)* operation. Note the size of the vector increases to 9, and the new elements are filled with a default value.

Algorithms

- Algorithms
  - STL provides several generic algorithms.
    - such as `find,replace,sort,...`
  - http://www.cplusplus.com/reference/algorithm/

```
#include <iostream>
#include <vector>
#include <algorithm>

void main() {
   int iarray[3] = {5,3,4};
   std::vector<int> ivec(3); ivec[0] = 5; ivec[1] = 3; ivec[2] = 4;

   std::sort(iarray, iarray + 3);
   std::sort(ivec.begin(), ivec.end());


  // std::sort can be applied to comparable objects
}
```

Example

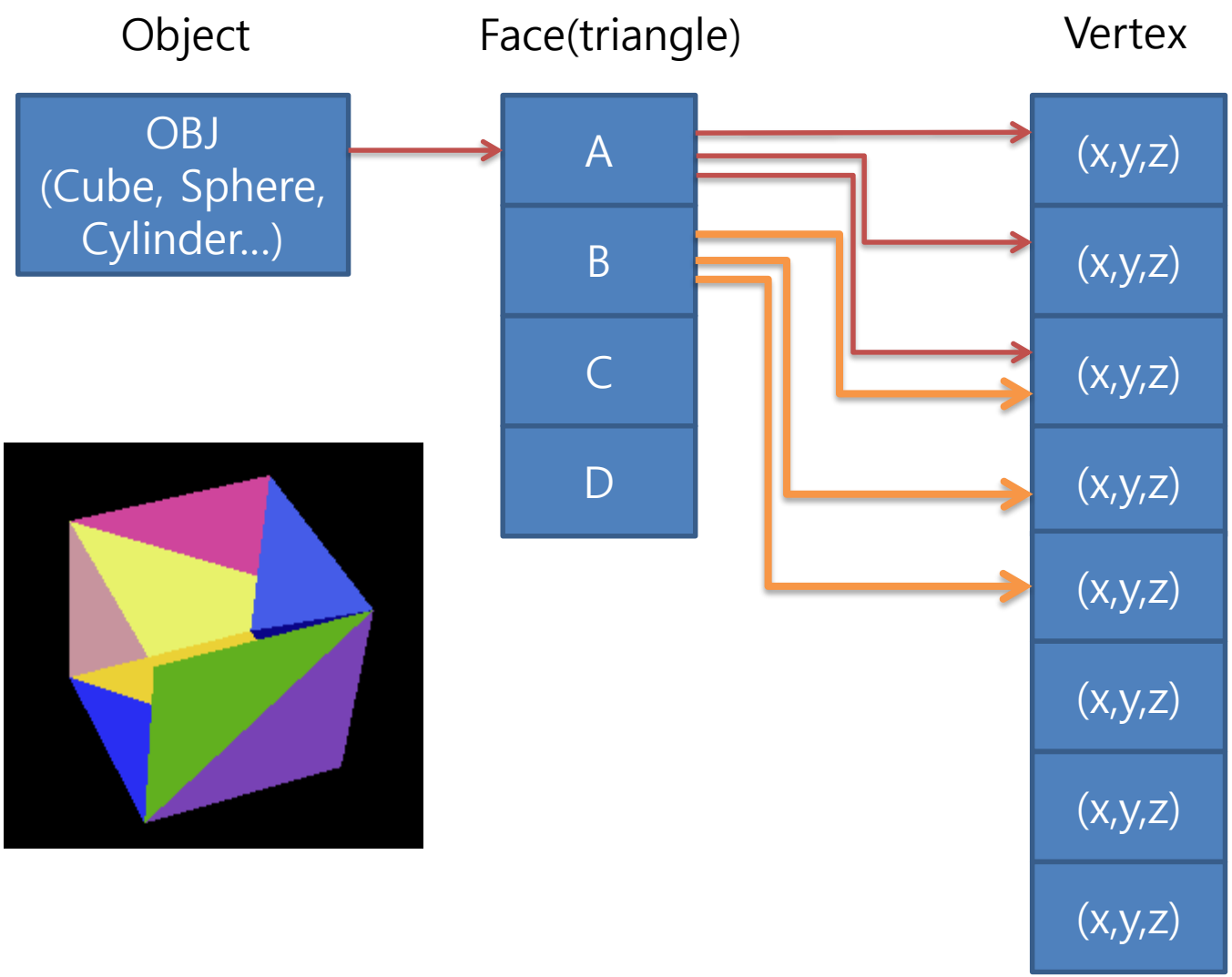
- Practice std::vector

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void main() {
   vector<float> ivec;
   ivec.push_back(3.14f); ivec.push_back(2.55f); ivec.push_back(1.22f);
   ivec.push_back(5.21f); ivec.push_back(2.25f); ivec.push_back(19.2f);

   cout << "Before pop_back() : " << ivec.size() << endl;
   ivec.pop_back();
   cout << "After pop_back() : " << ivec.size() << endl;

   cout << "### Before sorted ###" << endl;
   for(vector<float>::iterator it=ivec.begin();it!=ivec.end();++it)
      cout << *it << endl;

   sort(ivec.begin(), ivec.end());

   cout << endl << "### After sorted ###" << endl;
   for(vector<float>::iterator it=ivec.begin();it!=ivec.end();++it)
      cout << *it << endl;
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
Before pop_back() : 6
After pop_back() : 5
### Before sorted ###
3.14
2.55
1.22
5.21
2.25

### After sorted ###
1.22
2.25
2.55
3.14
5.21
계속하려면 아무 키나 누르십시오 . . .
```

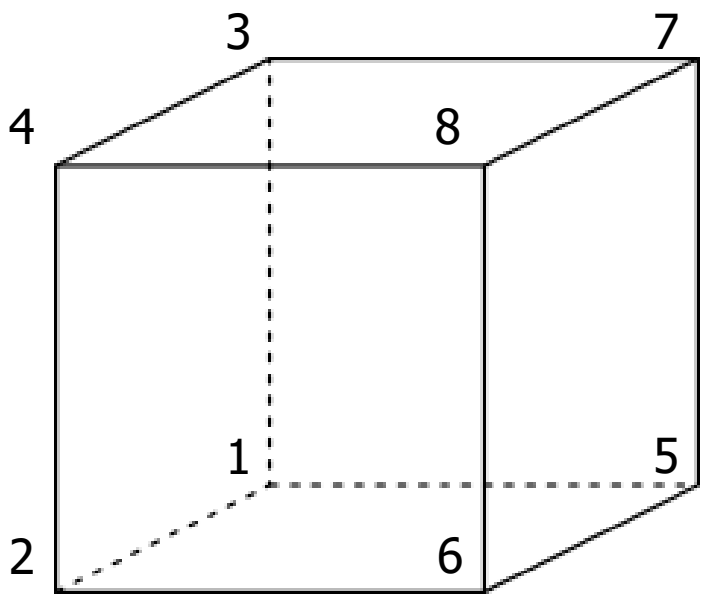
Object Structure



cube.obj

```
v 0.0 0.0 0.0
v 0.0 0.0 1.0
v 0.0 1.0 0.0
v 0.0 1.0 1.0
v 1.0 0.0 0.0
v 1.0 0.0 1.0
v 1.0 1.0 0.0
v 1.0 1.0 1.0
f 1 7 5
f 1 3 7
f 1 4 3
f 1 2 4
f 3 8 7
f 3 4 8
f 5 7 8
f 5 8 6
f 1 5 6
f 1 6 2
f 2 6 8
f 2 8 4
```

'v' : vertices with (x,y,z)



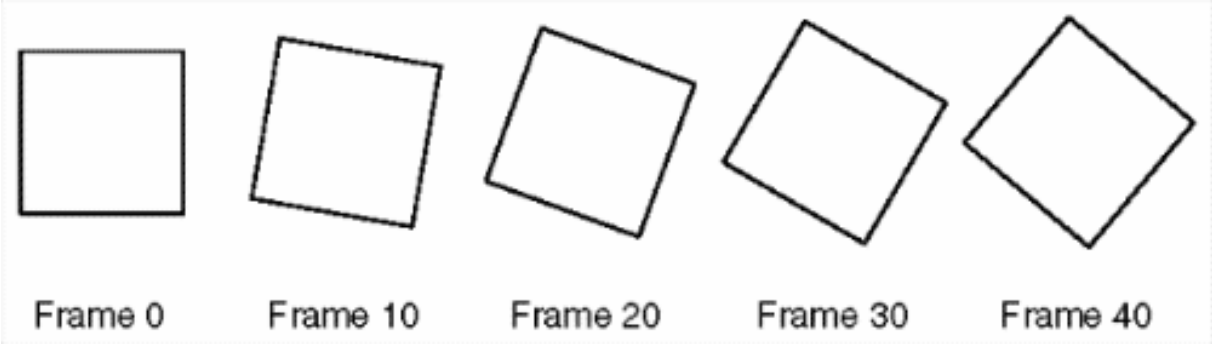
'f' : face (vertex index starts from 1)

Callback functions

- glutDisplayFunc(…)
- glutKeyboardFunc(…)
- glutSpecialFunc(…)
- glutMouseFunc(…)
- glutMotionFunc(…)
- glutIdleFunc(…)

glutIdleFunc(idle)

- GLUT program can perform background tasks or continuous animation when window system events are not being received if the idle function is enabled.



glutIdleFunc(idle)

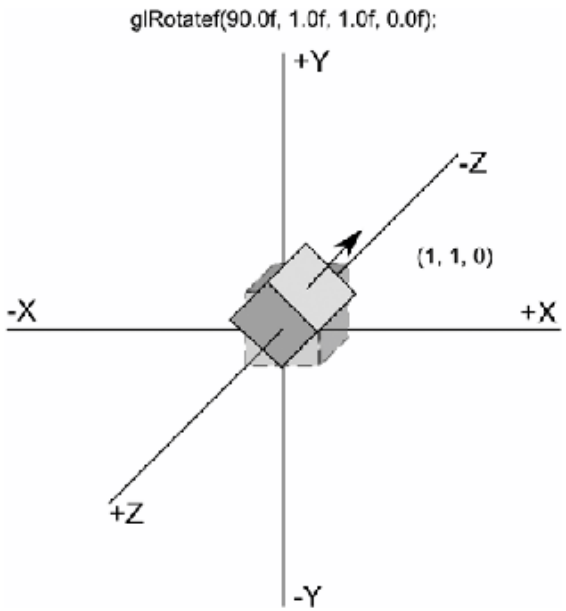
```
void renderScene() {
        glEnable(GL_DEPTH_TEST);
        // Clear Color and Depth Buffers
        glClear(GL_COLOR_BUFFER_BIT |
        GL_DEPTH_BUFFER_BIT);

        // Use the Projection Matrix
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        // Set the correct perspective.
        gluPerspective(45.0f, 1.0f, 0.1f, 100.0f);

        // Reset transformations
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        // Set the camera
        gluLookAt(8.0f, 5.0f, 2.0f,
        0.0f, 0.0f, 0.0f,
        0.0f, 1.0f, 0.0f);

        glRotatef(angle, 0.0f, 1.0f, 0.0f);

        Cube.draw();

        glutSwapBuffers();
}
```

```
void idle() {
        // change the angle
        …
        glutPostRedisplay();
}
```

```
void main(int argc, char **argv) {
        …

        // register callbacks
        glutDisplayFunc(renderScene);
        glutIdleFunc(idle);

        // enter GLUT event processing cycle
        glutMainLoop();
}
```

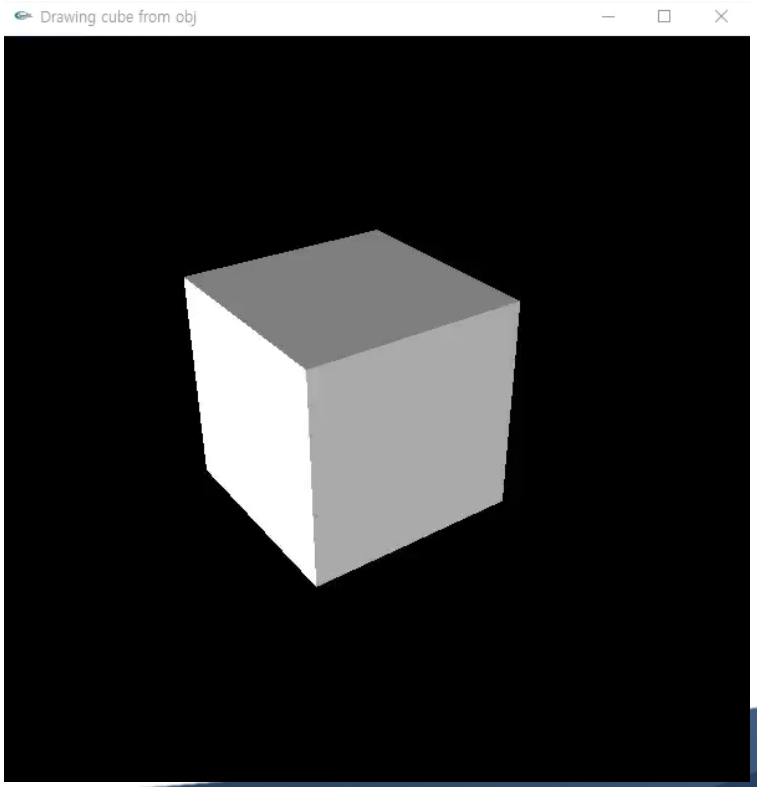
Rotation: glRotate

- glRotate{fd}(angle, axis_x, axis_y, axis_z);
  - angle: rotation angle in degrees in CCW
  - (axis_x, axis_y, axis_z): the rotation axis
- Example
  - glRotatef(135.0f, 0.0f, 1.0f, 0.0f); // rotation around y-axis
  - glRotatef(90.0f, 1.0f, 1.0f, 1.0f); // around axis (1, 1, 1)



Today's Mission

- Load data from OBJ file (cube)
  - Use STL vector
  - Define vertex, face, obj class
- Access member variables only through member functions
- Rotate the cube using idle function



Today's Mission

- Implementation
  - Define class
    - vertex
    - face
    - obj
  - Read data from obj file
  - Rotate the cube
    - Use idle callback function

vertex

- Member function
① Constructor
② setVertex(…)
③ getX()
④ getY()
⑤ getZ()

face

- Member function
① Constructor
② draw()

vertex

obj

- Member function
① Draw()
② ReadFromFile(…)

face

Today's Mission

- Implementation (details)
  - Define class
    - vertex
    - face
    - obj

```
class vertex {
public:
        /* Implement constructor, setter, getter */

private:
        float pos[3];
};
```

```
class face {
public:
        /* Implement constructor, draw function */

private:
        vertex vtx[3];
};
```

```
class obj {
public:
        void draw();
        void ReadFromOBJFile(string filename);

private:
        vector<face> vFace;
};
```

Today's Mission

- Implementation (details)
  - Define class
    - vertex
    - face
    - obj

```
void obj::draw() {
	/* Implement: draw all faces */
}

void obj::ReadFromOBJFile(string filename) {
	vector<vertex> vertices;
	ifstream input(filename);

	char sub;
	while (input >> sub) {
		if (sub == 'v') {
			/* Implement: read vertex data */
		}
		else if (sub == 'f') {
			/* Implement: read face data */
		}
	}
}
```

Today's Mission

- Implementation (details)
  - Read data from obj file

```
void main(int argc, char **argv) {
	/* Implement: Read data from OBJ file */

	// init GLUT and create Window
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
	glutInitWindowPosition(650, 300);
	glutInitWindowSize(WIDTH, HEIGHT);
	glutCreateWindow("Drawing cube from obj");

	// register callbacks
	glutDisplayFunc(renderScene);
	glutIdleFunc(idle);

	// enter GLUT event processing cycle
	glutMainLoop();
}
```

Today's Mission

- Implementation (details)
  - Rotate the cube
    - Use idle callback function

```
void idle() {
    /* Implement: Change the rotation angle */
}
```

```
void main(int argc, char **argv) {
    …

    // register callbacks
    glutDisplayFunc(renderScene);
    glutIdleFunc(idle);

    // enter GLUT event processing cycle
    glutMainLoop();
}
```