Inc to Engine

Class Design

- Node
  - Object :
    This class and its subclass include all the object that **can be detected when picking**(except Skybox and Camera , these two object don't need to be shown and detected **all the time in the world**)
    - Geometry
      - Cube **finished, including supporting of texture selecting**
      - Cone **just for example, not required**
      - Sphere **haven't add it yet, maybe later**
      - Domino **for Yue, also doesn't need to be designed like this**
    - Model : for generate the Object from .obj file **for Yiming**
      add a static address for the .obj file address and load it just for one time
      and later send a pointer into it and generate the model to the world
      it's better to scale the model initially to 1x1x1 for our need of restriction
    - Skybox **finished**
    - Camera **finished**
  - Light : **for Yiming**
    for generating the Light to the world, including illuminating other object near it (or all if directional light)
    - PointLight
    - DirectionLight
    - SpotLight
      You can start from the single light in the world, it easy to calc its shader effect. But there is a little difficult point for how to add light effect to the objects because **the shader** is belong to the object not light, maybe you can modify the Object and its subclasses' shader parameters for better use.
  - Transform **finished actually not used now**
    - Translate
    - Rotate
    - Scale
  - Coordinate "finished"

Show the Coordinate for x-z

- Window **serve as a controller**

How to play

Press C to show the coordinate
Noticing that all the objects ,the orginal position is (0,0,0), with Lower left corner alignment (**not center!!!** , you can obey this rule in the Model class design)

Move your mouse to see the world
Use W,A,S,D,E,Q to go forward,left,backward,right,up,down
Press 0 to reset the placing
Press 1 to use the Cube placing
Press 2,3,4,…. for the later use

When choosing a placing (meaning that you press 1,2,3,…. not 0)
Press left button to place it
Press right buttono to delete it
Press I O to change the textures(prev, next), it's okay use only one key, the seq queue is cyclic automatically
what the position of your placement is shown with lightly red color(just like minecraft)
Press 0 to reset the placing(means that you place nothing when press mouse button)
Forget it about some wrong placements if you feel strange when placing someplace wrong.

How to create objects

```
Cube cube = new Cube(idCount++, 1, style); // to add the Cube in th
e world ,orginal in (0,0,0)
// there are two construct for Cube,
// this one, the first is the id, when creating one object, just ad
d one
// the second is the size, actually we use one often,
// the third is the style, meaning the textureId, the texture
// there is another one construct in the Cube.h for color
cube->setPosition(npos.x, npos.y, npos.z);
// set the cube Position
objectList.push_back(cube);
// This objectList contains all the objects in the world, except sk
ybox, camera, and later light.
// just for the pick detection and later object collision.
```

The others are almost same

```
skybox = new Skybox(idCount++, 1000, &faces); // Skybox id,size, fa
ces
currentCam = new Camera(idCount++, glm::vec3(0, 1, 10), glm::vec
3(0, 0, 0), glm::vec3(0, 1, 0));
// camera id, pos, lookat ,up
coordinate = new Coordinate(idCount++, 100);
// coordinate
```

See the Object and Cube class for detailly understand my design. It freely for you to modify (especially for lighting use)

Check the window class carefully for understanding the initialization and some useful functions
Like:

```
static void placeObject(int type, int *style);
static Object* testForCollision(int *face);
// etc.
```

check the Util,h and cpp carefully, which defined many value for loading
It's important that when you declare some variables in Util.cpp like "Shader" for global use (be sure in cpp not in h. , otherwise there will be wrong), you should **extern** it in other file like :

```
// In util.cpp
GLuint Shader_Geometry;
GLuint Shader_Skybox;
GLuint Shader_Coordinate;
```

And:

```
// In window.cpp
extern GLuint Shader_Geometry;
extern GLuint Shader_Skybox;
extern GLuint Shader_Coordinate;
```

Otherwise it will cause LNK2005 error and the linking will not be the same in different class.

Good luck.