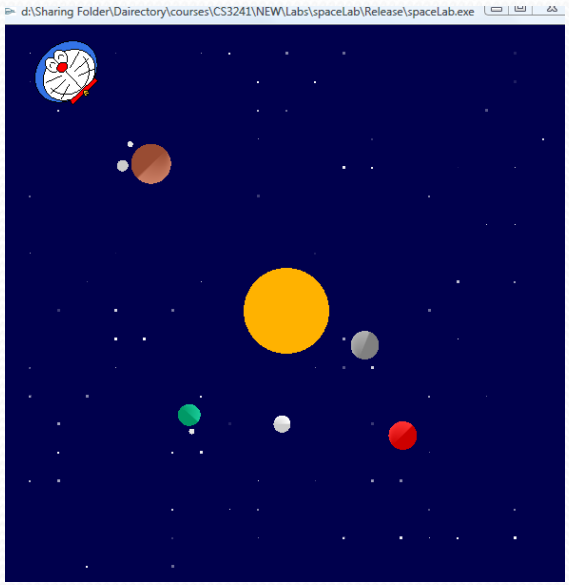
HW2: The World is Round

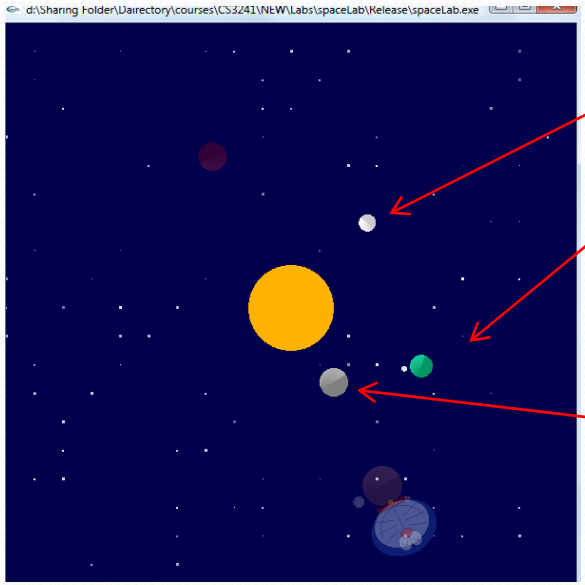
Lab 2: The World is Round

Solar System Mode



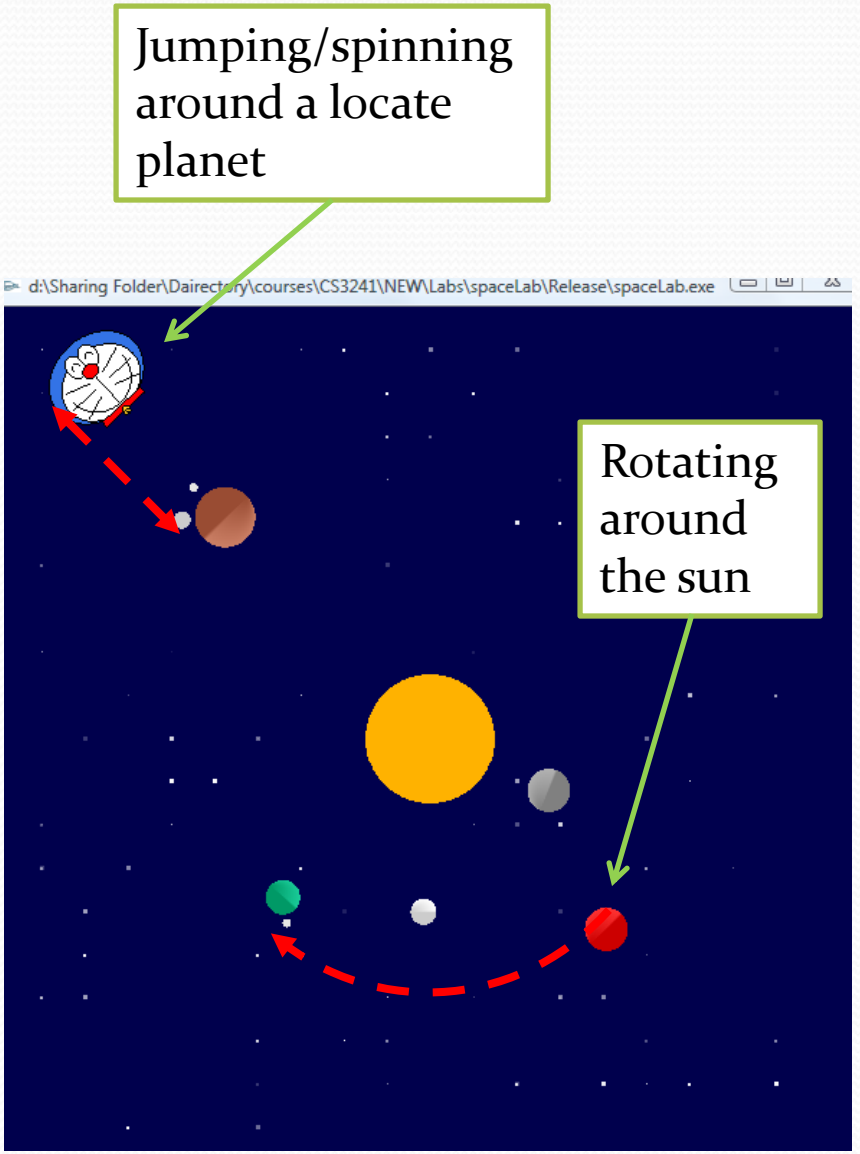
Toggle by the key "T"

Clock Mode (5:07:20)



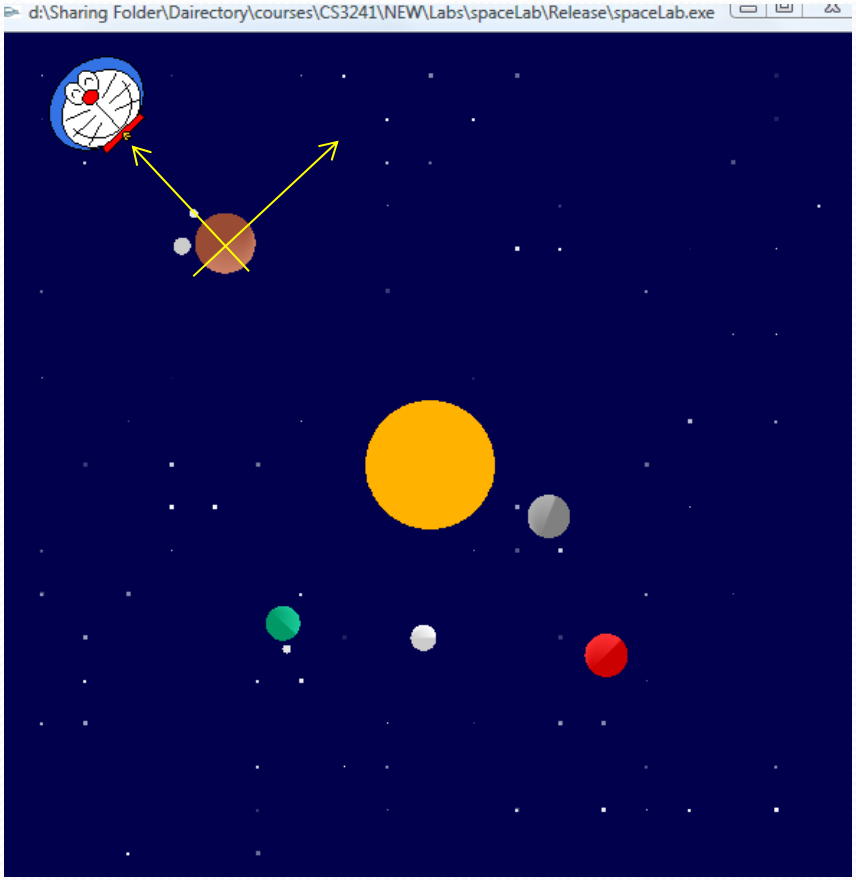
Solar Mode

- Requirement: some motions with respect to local frames
  - E.g. a moon spinning around a planet and Doraemon jumping on a planet
- You are allowed to reuse your code in Lab 1
- Even non-circular orbits are allowed
  - Elliptic, hyperbolic, etc…
  - But not linear



Solar Mode: Aim

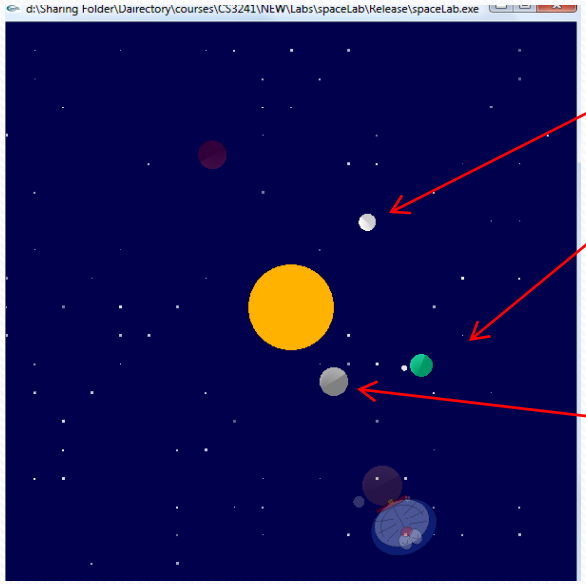
- Understanding the usage of local reference frames
  - You may even try more than one layer of reference frame



Clock Mode

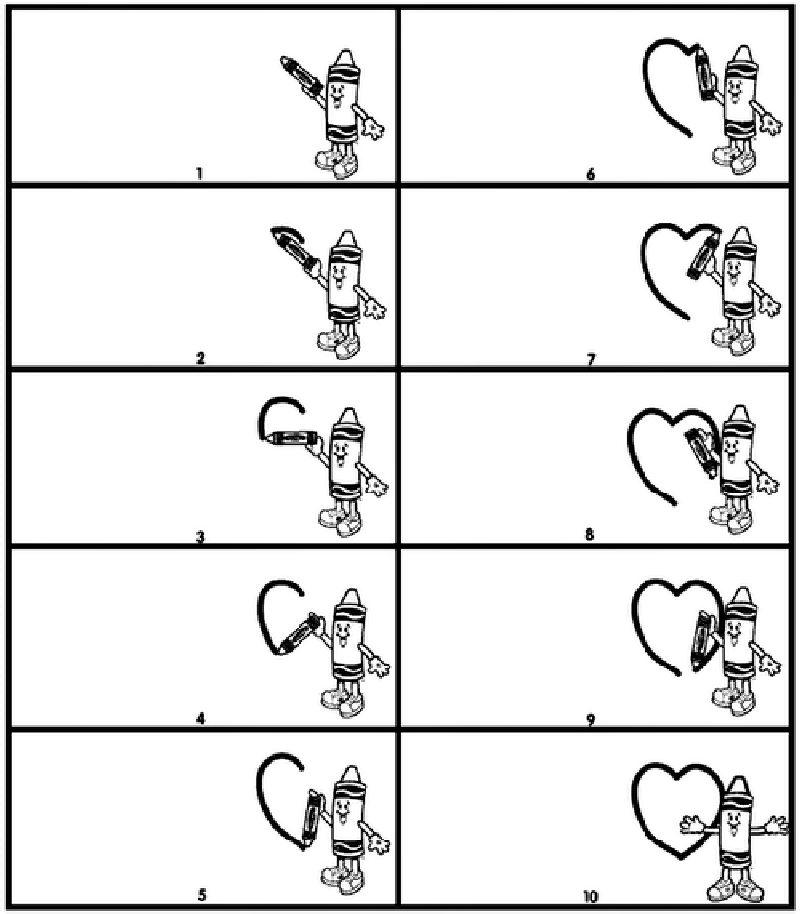
- Requirement
  - Moving “clock arms” according to the time
    - Showing hours, minutes and seconds
  - You can dim/add/subtract objects to/from the solar mode

5:07:20



Animation

- Displaying different frames according to time
- However, GLUT main loop sleeps if there is no event
  - Thus, the display function will not redraw a new picture/frame
- Need to register a function to wake the program when idling



glutIdleFunc

- Use `glutIdleFunc(myIdle)` to register a function `myIdle()`
  - So that `myIdle()` will be called whenever the program is idle
- A very simple idle function :

```
void myIdle(void)
{
        glutPostRedisplay();
}
```

- This function posts a “redisplay” event (thus, call your display function) whenever the program is free

Try This

```
void myIdle(void)
{
  glutPostRedisplay();
}
void myDisplay()
{
  glPushMatrix();
  time_t seconds = time (NULL);
  struct tm * timeinfo = localtime(&seconds);
  double angle = 360-(float)timeinfo->tm_sec/60*360;
  glRotatef(angle,0,0,1);
  drawSomething();
  glPopMatrix();
}
```

Clock Mode

- Get number of seconds from 00:00:00 1 Jan 1970
  - `time_t time(NULL);`
- Then use the function `localtime` to convert to local time, e.g. Calculating the angle for the second needle

```
time_t seconds = time (NULL);
struct tm * timeinfo = localtime(&seconds);
double angle = 360-(float)timeinfo->tm_sec/60*360;
```

- But if you want something up to millisecond, you need some other timing function of C++

Transparency

Transparency

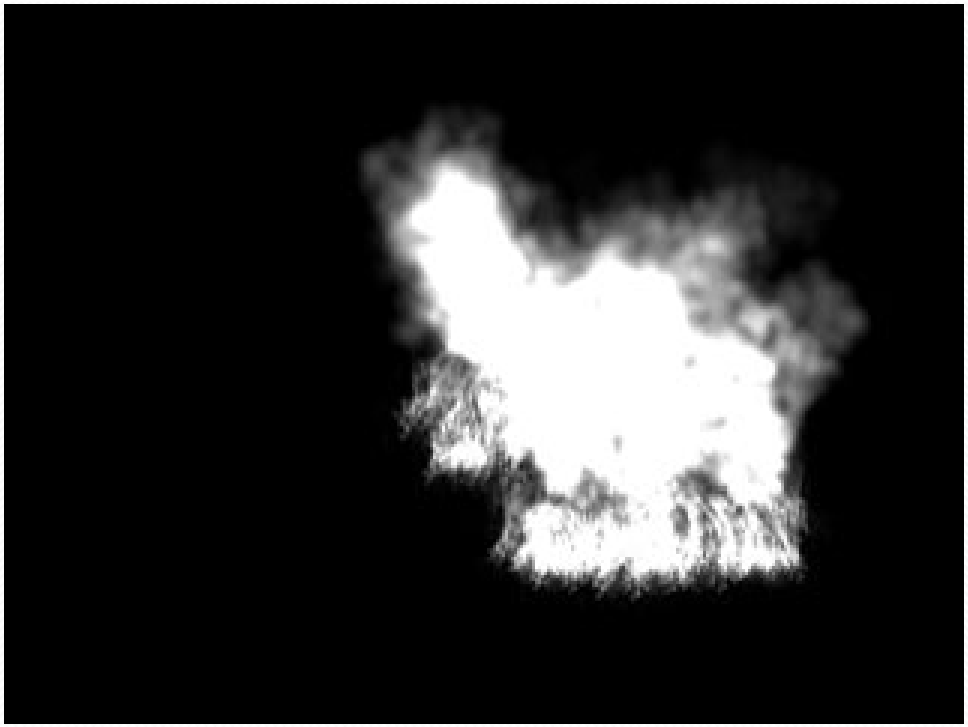
- Use of alpha channel
  - 32-bit graphics systems contain four channels:
    - three 8-bit channels for red, green, and blue (RGB) and one 8-bit alpha channel
  - Rendering overlapping objects that include an alpha value is called alpha blending

Transparency

- Use of alpha channel

Transparency

- Use of alpha channel



Transparency

- Use of alpha channel

Transparency

- Use of alpha channel
  - Color transparency
    - `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)`
    - `glEnable(GL_BLEND)`
    - `glColor4f(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha)`
  - What's more?
    - Google for OpenGL texture transparency, if you are interested

Transparency

- Try out

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glEnable(GL_BLEND);
glColor4f(0,0,1,0.5);
glBegin(GL_POLYGON);
   glVertex3f(-2,-1,0);
   glVertex3f(2,-1,0);
   glVertex3f(2,1,0);
   glVertex3f(-2,1,0);
glEnd();
glColor4f(0,1,1,0.4);
glBegin(GL_POLYGON);
   glVertex3f(-1,-2,0);
   glVertex3f(1,-2,0);
   glVertex3f(1,2,0);
   glVertex3f(-1,2,0);
glEnd();
```