Variable Frame Rate in OpenGL

Francesc Sebé

‘Computació gràfica i multimèdia’

Escola Politècnica Superior

Universitat de Lleida

Animation

- Moving scenes are shown as a sequence of frames
  - The **frame rate** (frames/second) is a fundamental parameter
    - Constant frame rate
    - Variable frame rate

Constant Frame Rate

- Achievable using a timer

```
void Timer(int n)
{
    glutTimerFunc(250, Timer, 0);          // 4 frames/sec

    // Update  Object  Position  for  Next  Frame

    glutPostRedisplay();          // Generate a “display” event
}
```

Constant Frame Rate

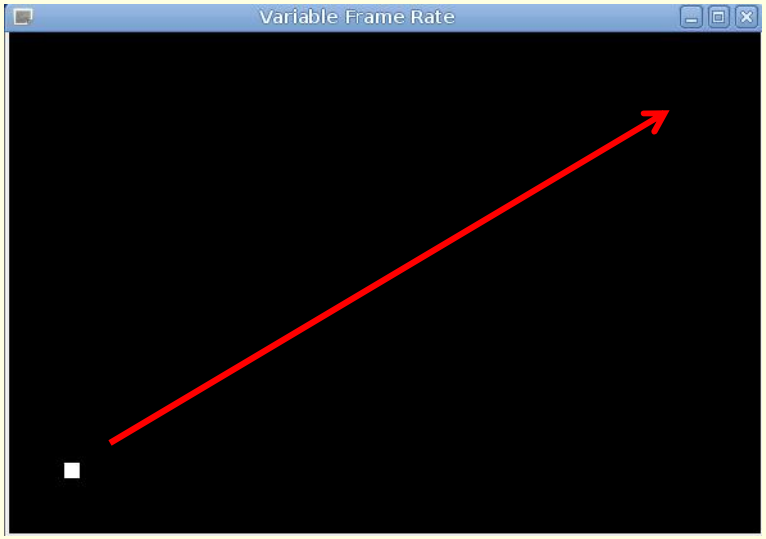
- Time between two frames must be enough to
  - Update data for next frame
  - Draw the next frame
- Otherwise, the animation will slow down
- The permitted frame rate varies from a machine to another
  - Which one do we select?

Variable Frame Rate

- Our animation frame rate will be the highest possible in the current machine
- It may vary depending on
  - How busy the computer is
  - The complexity of frame composition and drawing

VFR: example

- Simple application
  - A square moving across the screen each time a key is pressed
  - The movement takes 1 second



VFR: example

```
class particle {
  float x,y;                    //-- Current position
  float vx,vy;                  //-- Velocity vector
  int state;                    //-- QUIET or MOVE

  long time_remaining;

public:

  particle();
  void set_position(int x,int y);
  void init_movement(int destination_x,int destination_y,int duration);
  void integrate(long t);
  void draw();
};
```

VFR: example

```
particle::particle()
{
  state=QUIET;
}



void particle::set_position(int x,int y)
{
  this->x = x;
  this->y = y;
}
```

VFR: example

```
void particle::init_movement(int destination_x,int destination_y,int duration)
{
  vx = (destination_x - x)/duration;
  vy = (destination_y - y)/duration;

  state=MOVE;
  time_remaining=duration;
}
```

Compute the velocity vector

VFR: example

```
void particle::integrate(long t)
{
  if(state==MOVE && t<time_remaining)
    {
      x = x + vx*t;
      y = y + vy*t;
      time_remaining -= t;
    }
  else if(state==MOVE && t>=time_remaining)
    {
      x = x + vx*time_remaining;
      y = y + vy*time_remaining;
      state=QUIET;
    }
}
```

Update particle position according to the elapsed time (t)

VFR: example

```
void particle::draw()
{
  glColor3f(1,1,1);
  glBegin(GL_QUADS);
  glVertex2i(x-6,y-6);
  glVertex2i(x+6,y-6);
  glVertex2i(x+6,y+6);
  glVertex2i(x-6,y+6);
  glEnd();
}
```

The particle is drawn as a square

VFR: example

```
void display()
{
  glClearColor(0,0,0,0.0);
  glClear(GL_COLOR_BUFFER_BIT);

  square.draw();

  glutSwapBuffers();
}
```

The particle knows its position.

VFR: example

```
void keyboard(unsigned char c,int x,int y)
{
  square.set_position(50,50);
  square.init_movement(WIDTH-50,HEIGHT-50,1000);

  glutPostRedisplay();
};
```

VFR: example

```
void idle()
{
  long t;
  t=glutGet(GLUT_ELAPSED_TIME);

 square.integrate(t-last_t);

last_t = t;

 glutPostRedisplay();
}
```

Milliseconds since glutInit was called.

Time since the last position update.