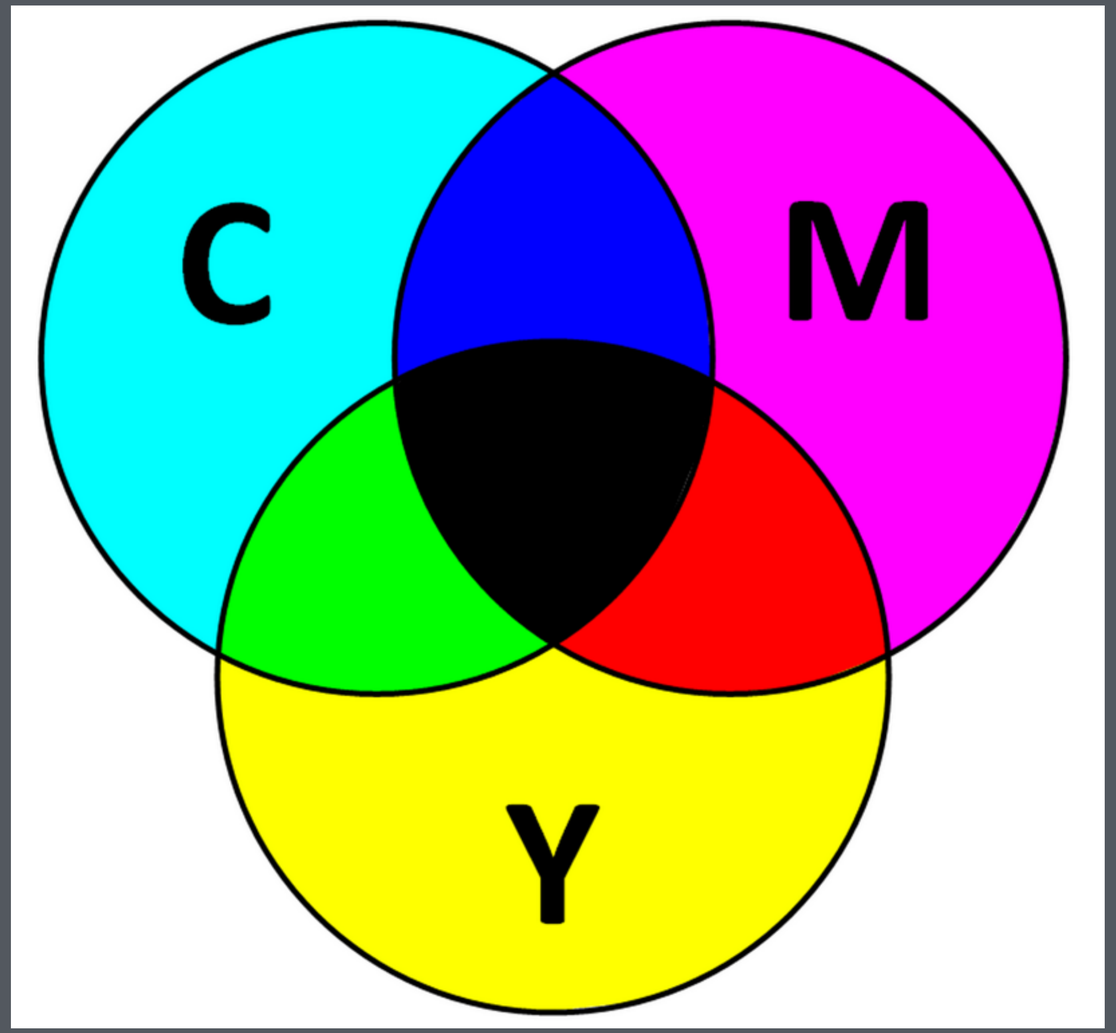
Graphics Foundations

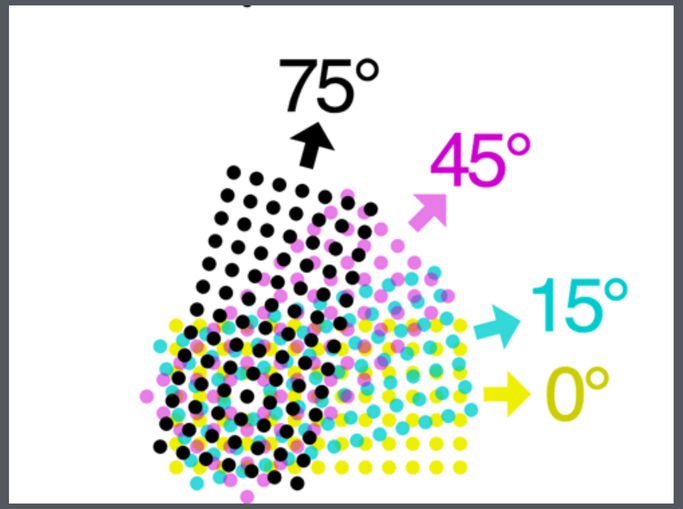
Part 2

Colors in computer graphics.

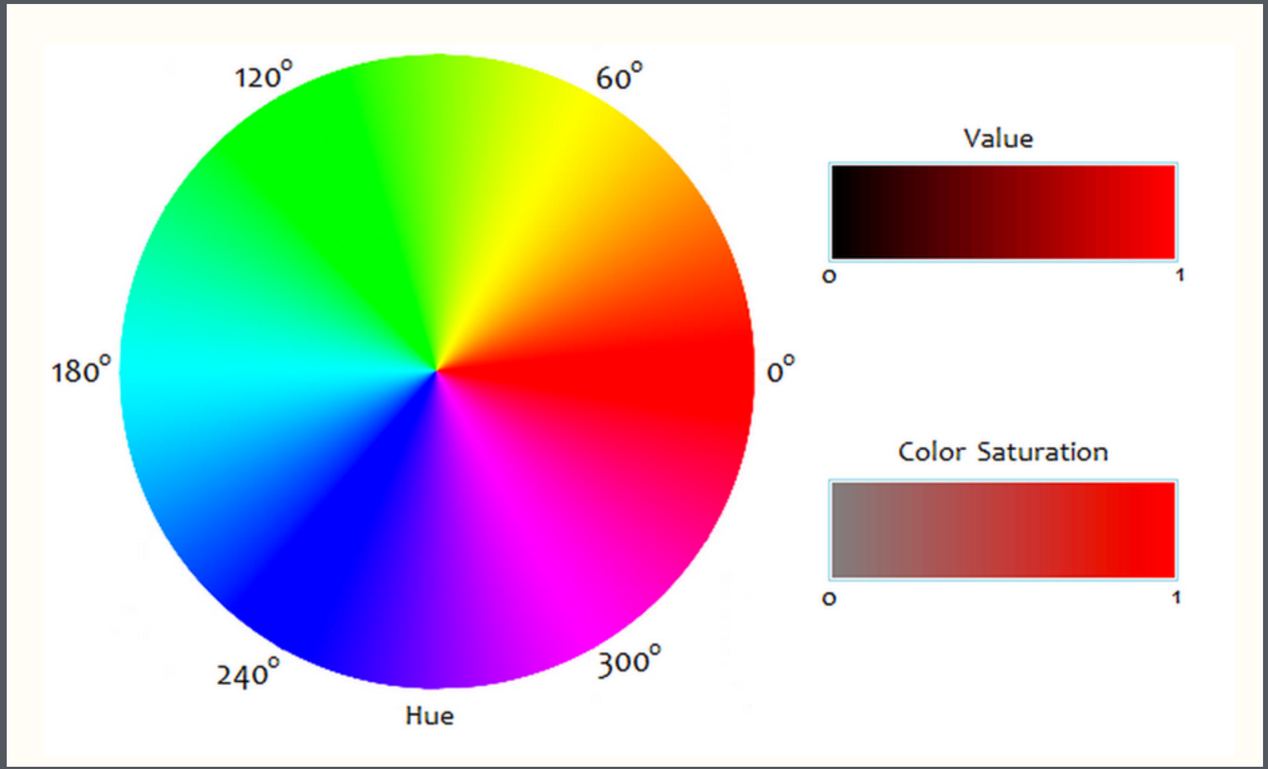
How can **color** be represented?

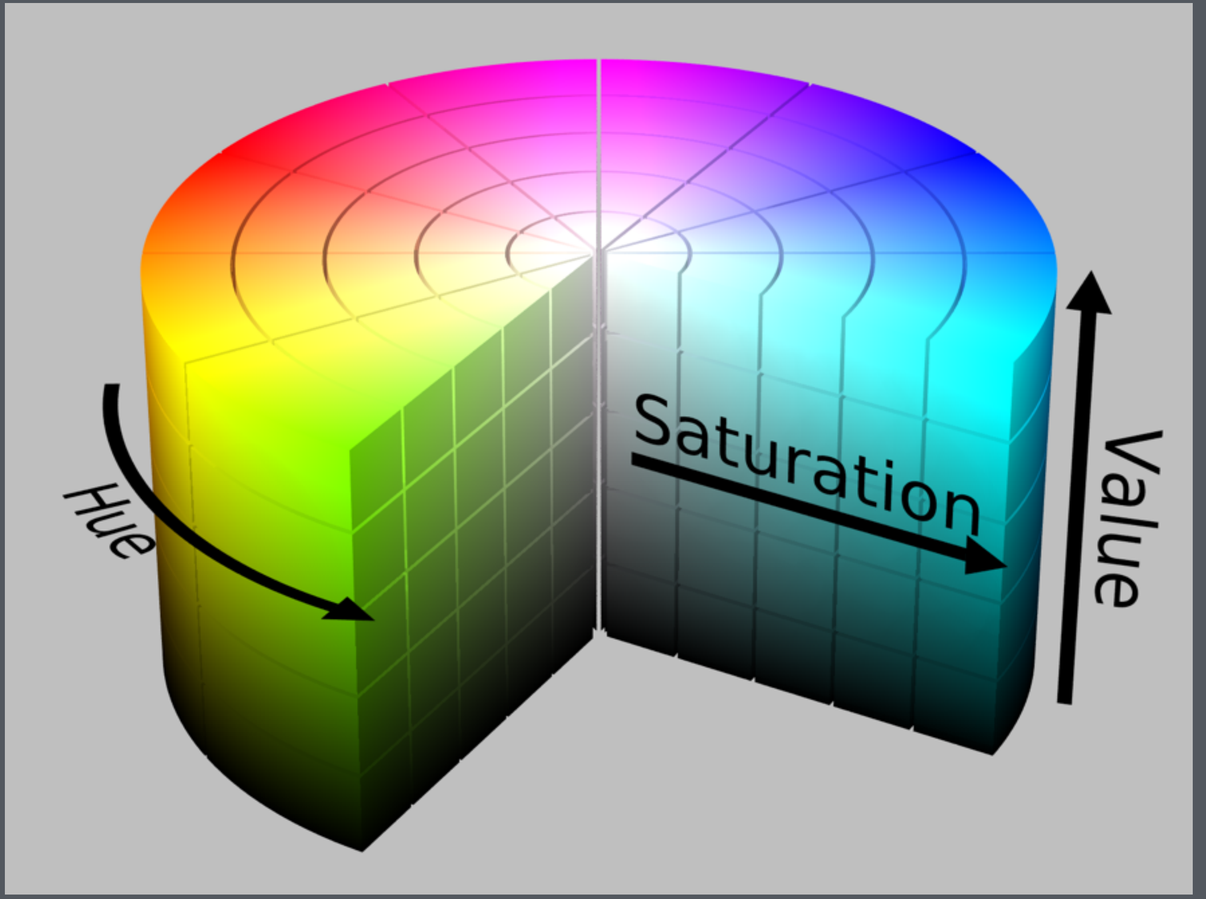
The **CMYK** Model





The **HSV** Model





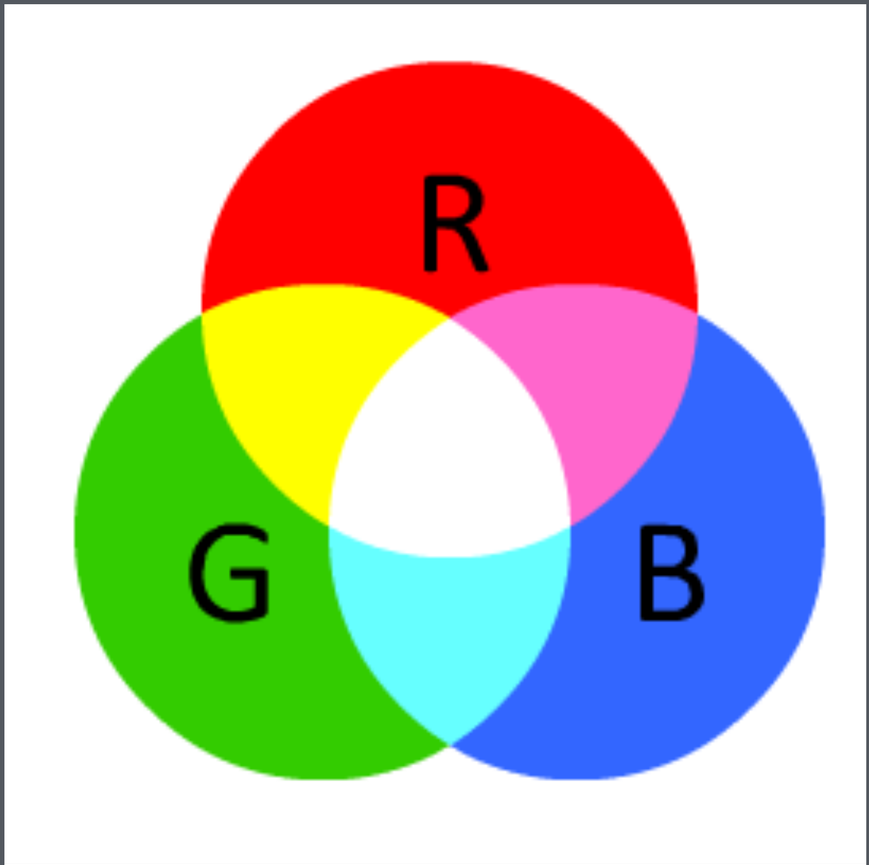
The **YUV** Model

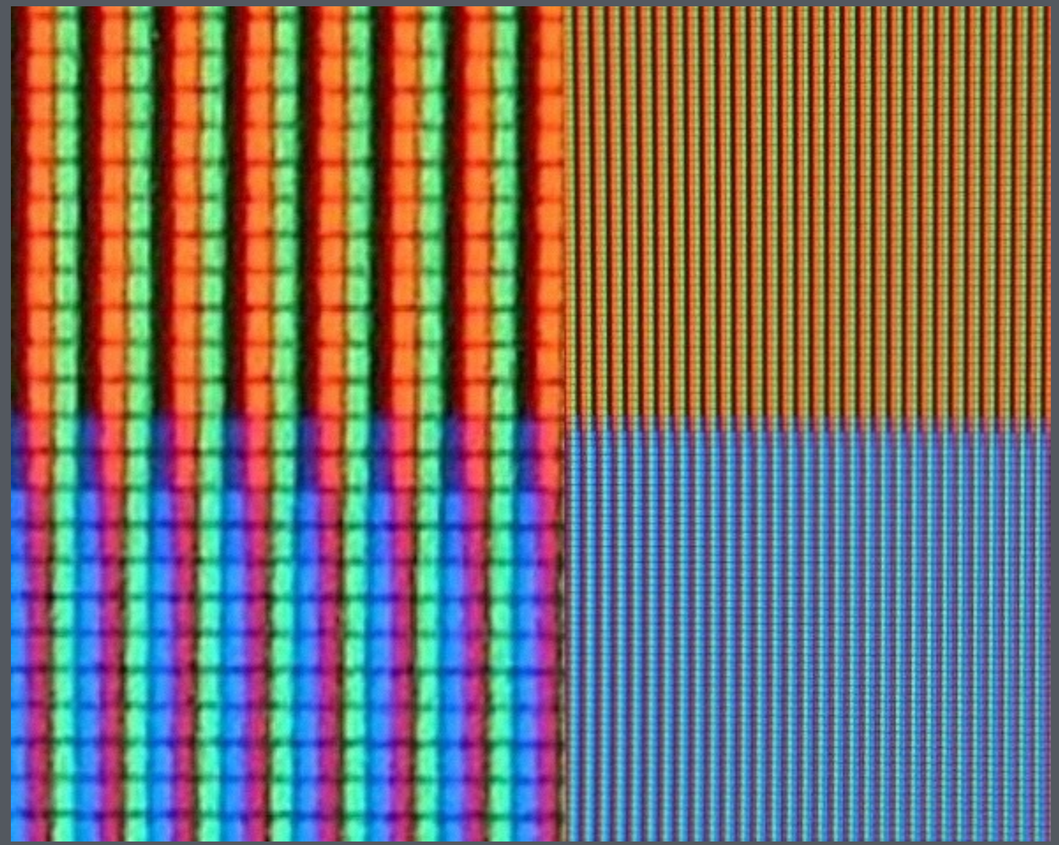
Luminance

Chrominance

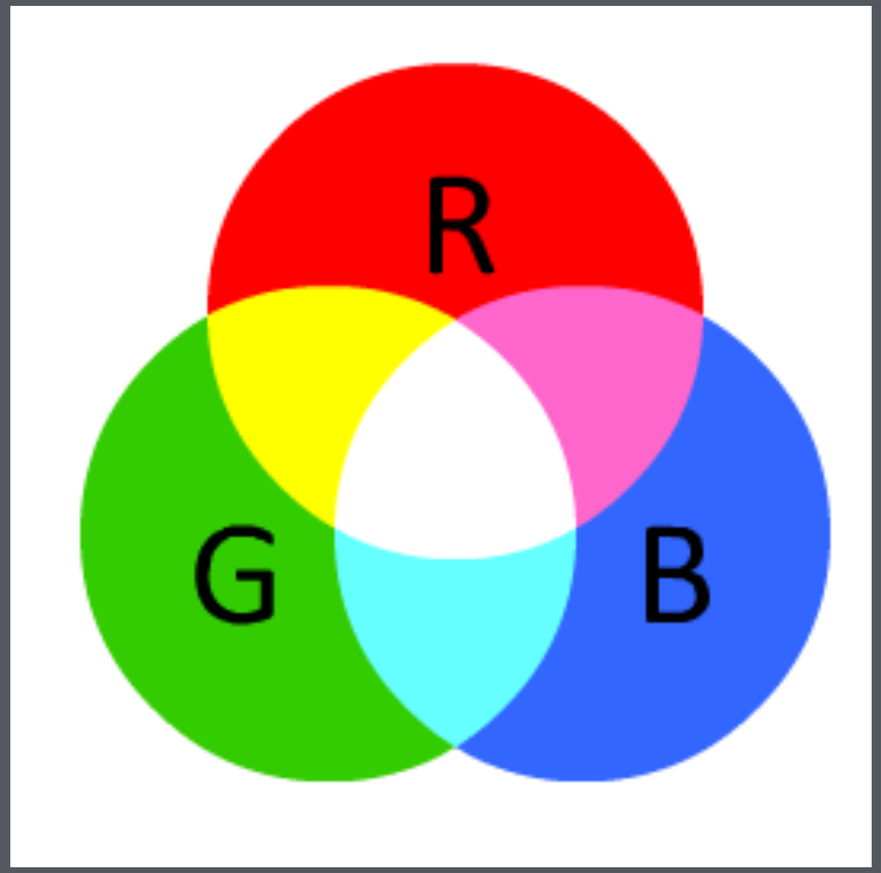
Both

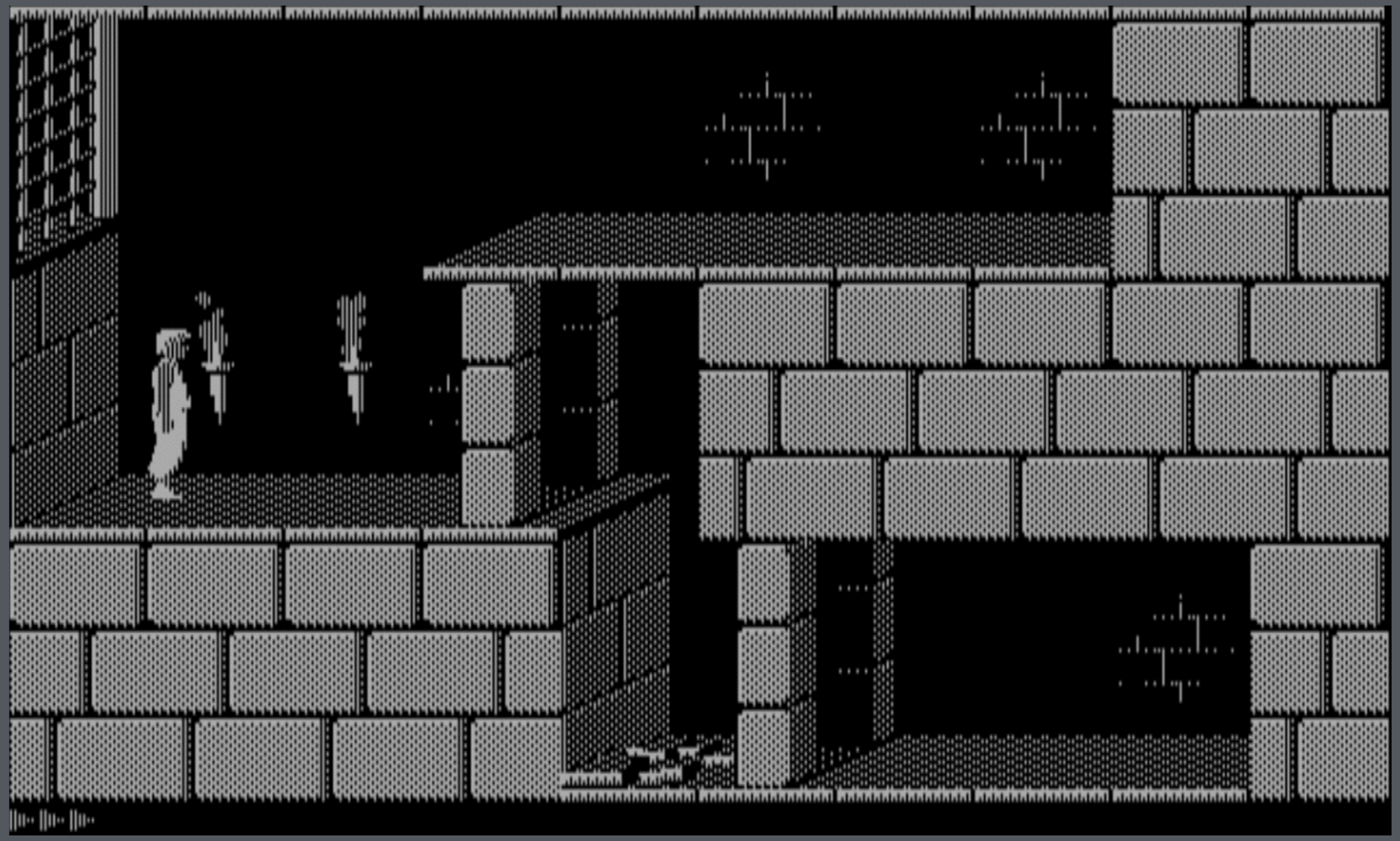
The **RGB** Model



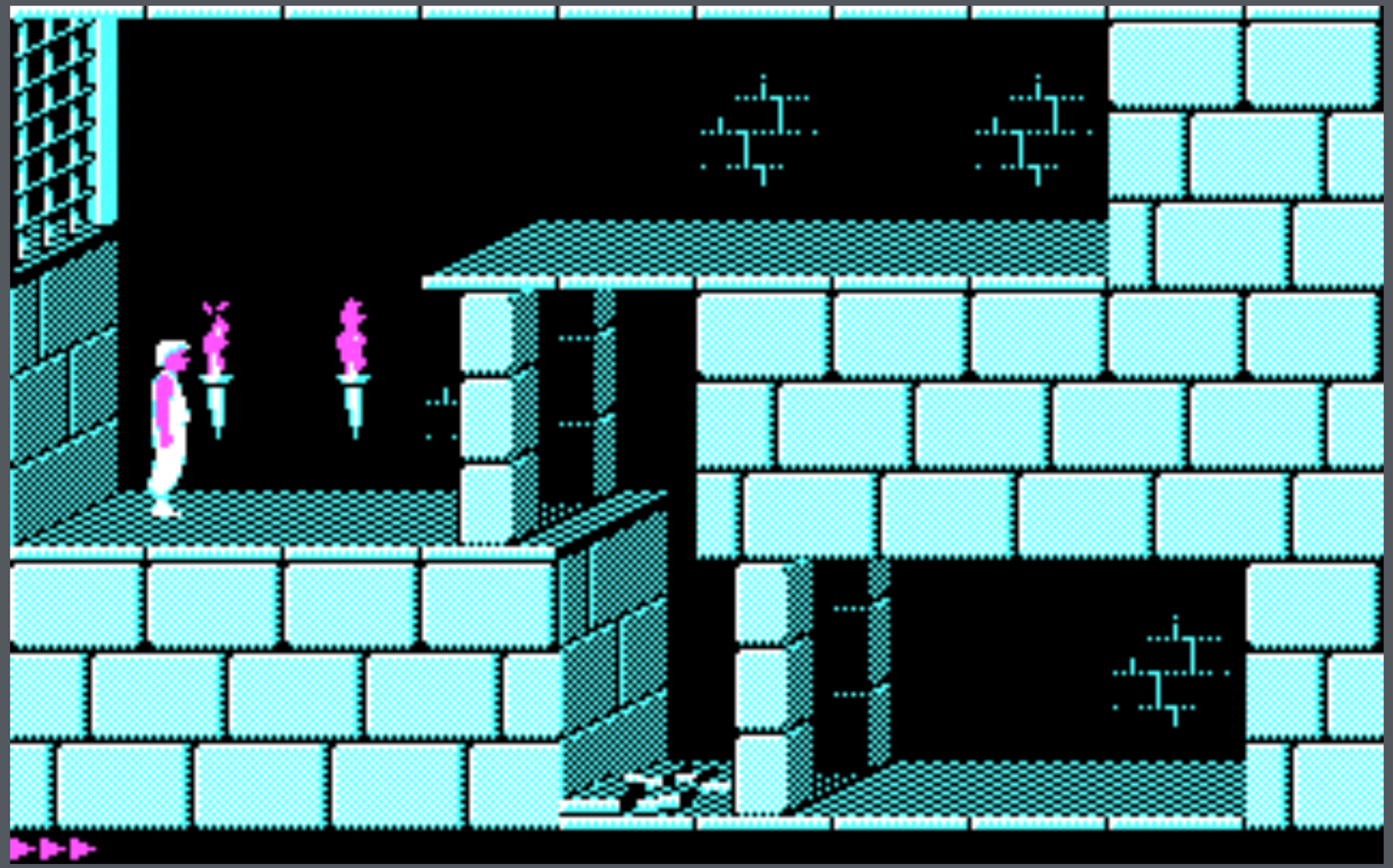


The **RGB Model** is most common in computer graphics.



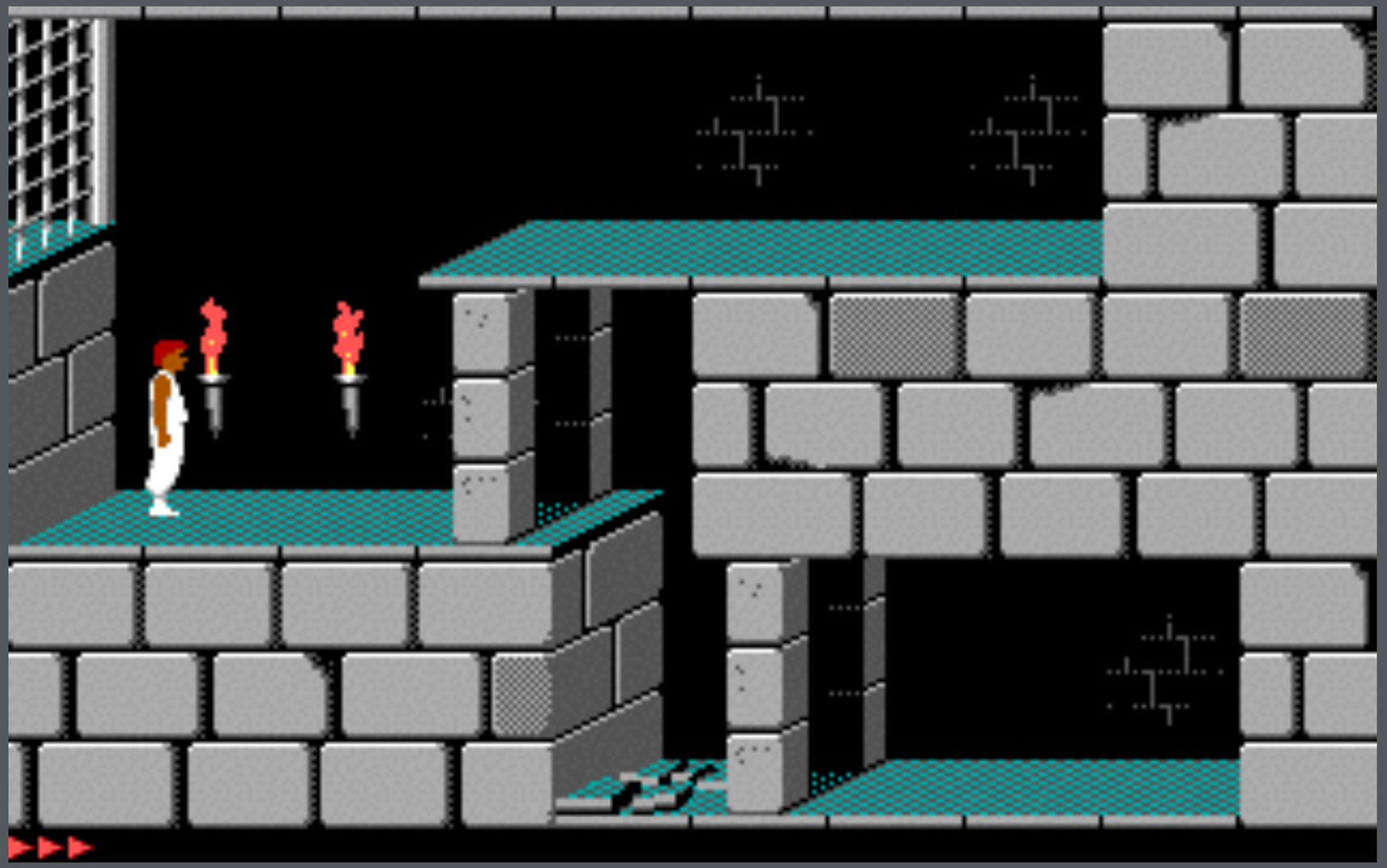


1-bit per pixel. Can either be **on** or **off**.



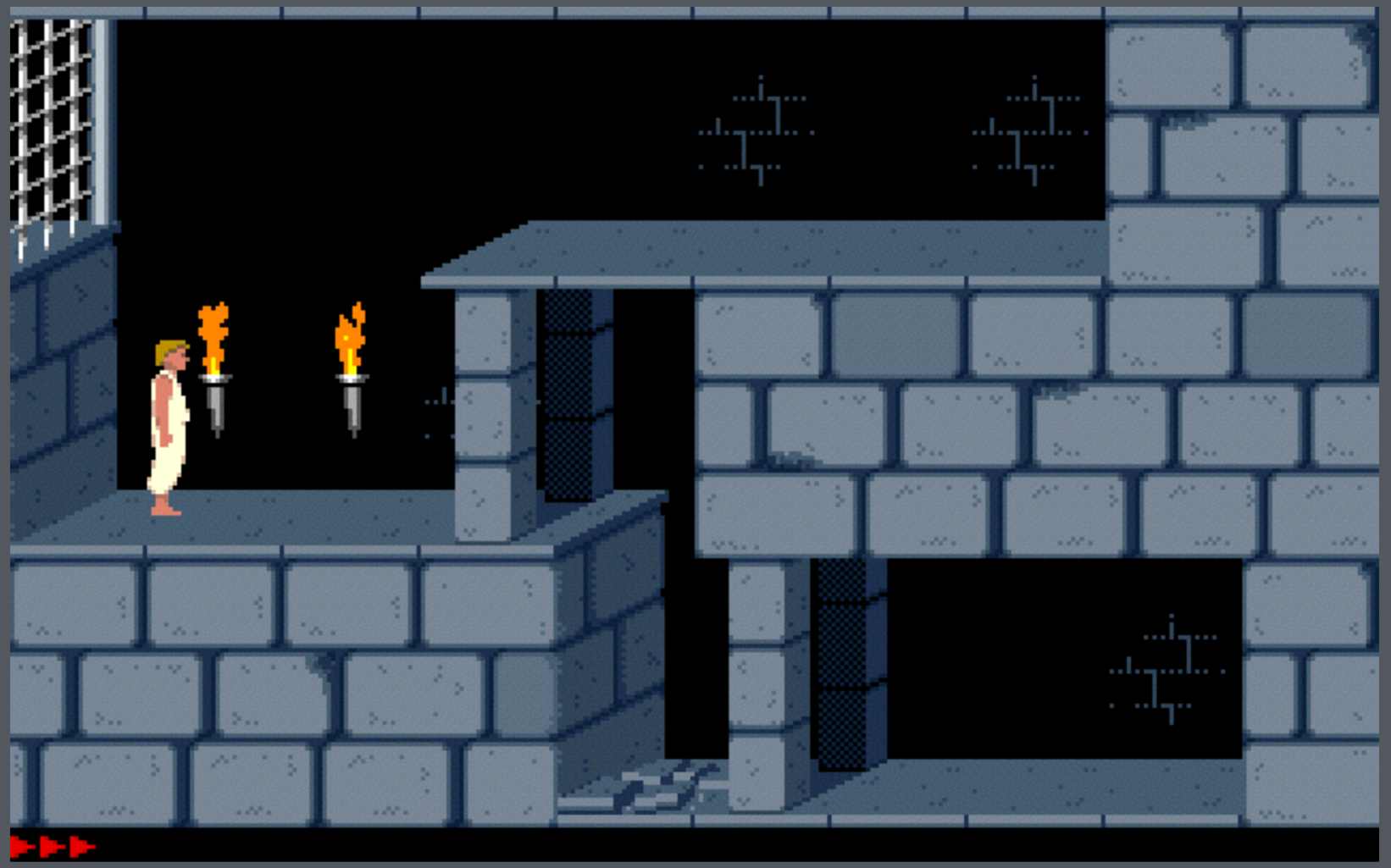
2-bit per pixel. Can be **4 different colors**.

This is called **indexed color**.

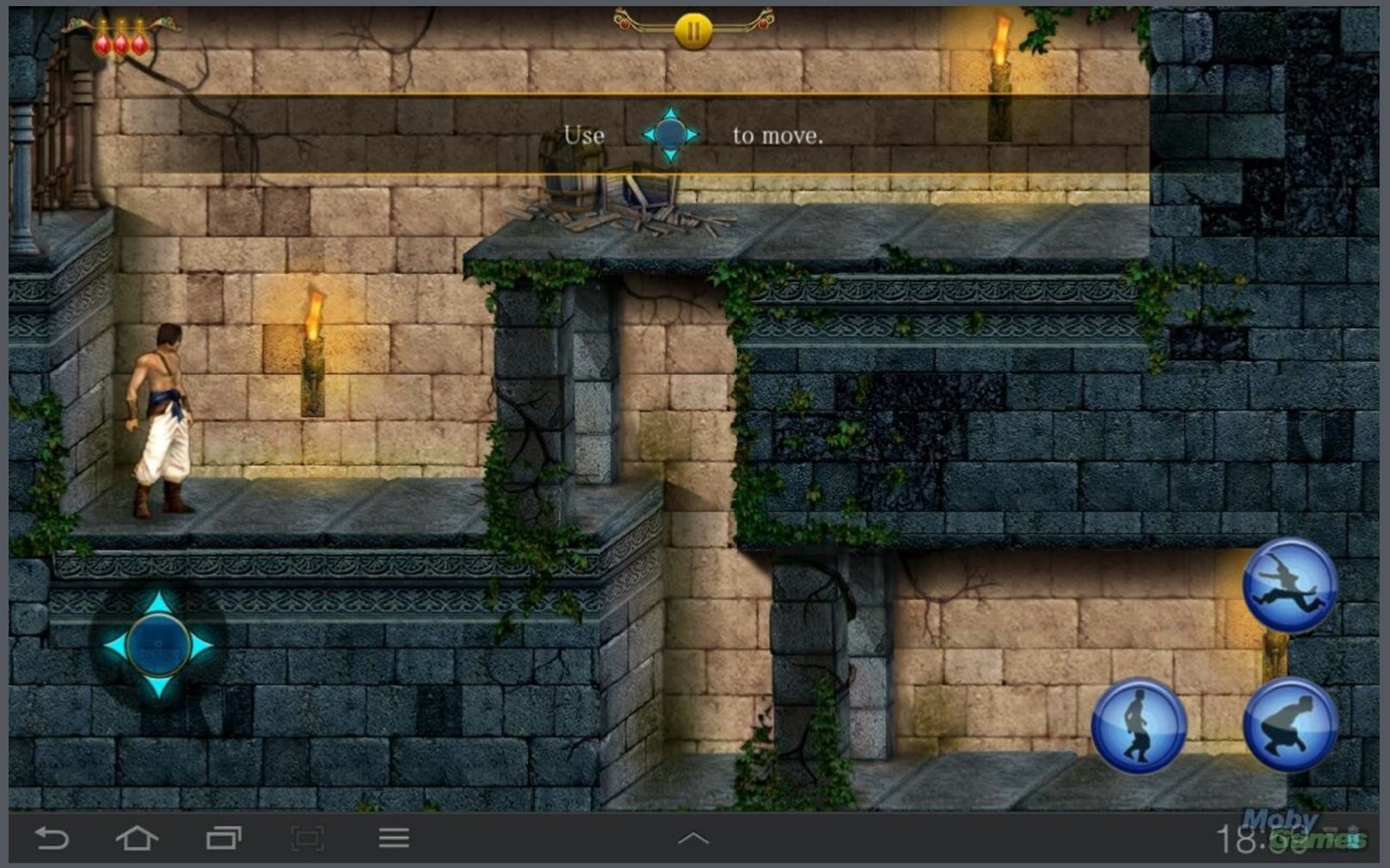


4-bit per pixel. Can be **16 different colors**.

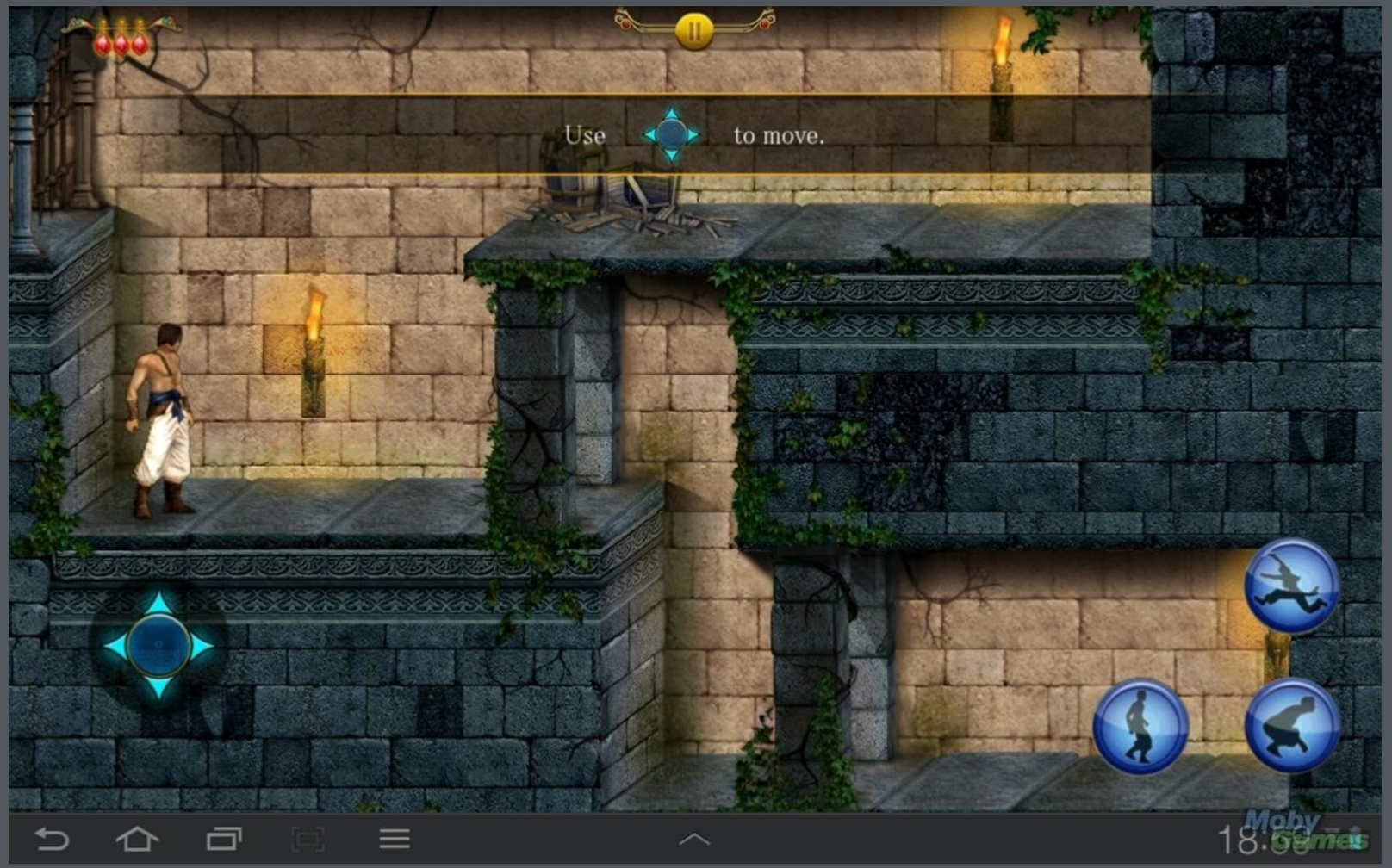
The color is still **indexed**.



8-bit per pixel (1 byte = 1 pixel). Can be **256 different colors**.
The color is still **indexed**.



24-bit per pixel. Can be any color we want (16777216 possible colors). This is called **RGB color**.



PIXEL PIXEL PIXEL PIXEL PIXEL

32-bit per pixel. Any color plus a **transparency level**.

This is called **RGBA color**.

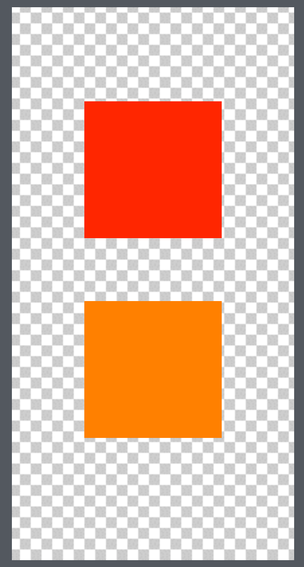
PIXEL PIXEL PIXEL PIXEL

HDR. **48-bit** per pixel or **96-bit** per pixel. RGB channels with **16 or 32 bits per channel**.

Colors in **OpenGL**.

RGB and **RGBA** colors as
0.0 - **1.0** floating point channels.

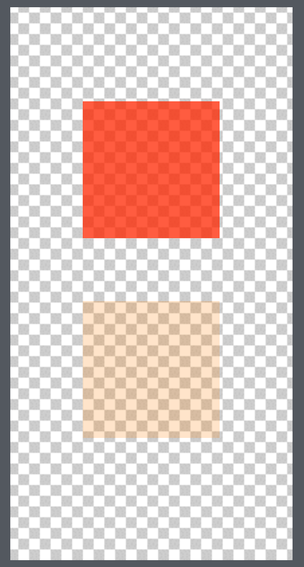
RGB



1.0, 0.0, 0.0

1.0, 0.5, 0.0

RGBA



1.0, 0.0, 0.0, 0.7

1.0, 0.5, 0.0, 0.2

Clearing the screen.

```
void glClearColor (GLclampf red, GLclampf
green, GLclampf blue, GLclampf alpha);
```

Sets the **clear color** of the screen.

```
glClearColor(0.4f, 0.2f, 0.4f, 1.0f);
```

```
void glClear (GLbitfield mask);
```

Clears the screen to the set clear color.

```
 glClear(GL_COLOR_BUFFER_BIT);
```

Textures and **images**.

SDL Surfaces

```
SDL_Surface *surface = IMG_Load("myimage.png");
```

```
typedef struct SDL_Surface
{
    int w;          // width of the image
    int h;          // height of the image
    int pitch;      // size of a row in bytes
    void *pixels;   // pointer to pixel data

    // …other data

} SDL_Surface;
```

Textures in OpenGL

Creating a texture

```
void glGenTextures (GLsizei numTextures, GLuint *textures);
```

Generates a new OpenGL **texture ID**.

```
GLuint textureID;
glGenTextures(1, &textureID);
```

Binding a texture

```
void glBindTexture (GLenum target, GLuint texture);
```

Bind a texture to a texture target.

```
glBindTexture(GL_TEXTURE_2D, textureID);
```

Our texture target is always going to be **GL_TEXTURE_2D**

Setting **texture data**

```
void glTexImage2D (GLenum target, GLint level, GLint
internalformat, GLsizei width, GLsizei height, GLint
border, GLenum format, GLenum type, const GLvoid *pixels);
```

Sets the **texture data** of the specified **texture target**. Image format must be **GL_RGBA for RGBA images** or **GL_RGB for RGB images**.

On the Mac, images will load as **GL_BGRA** or **GL_BGR** format.

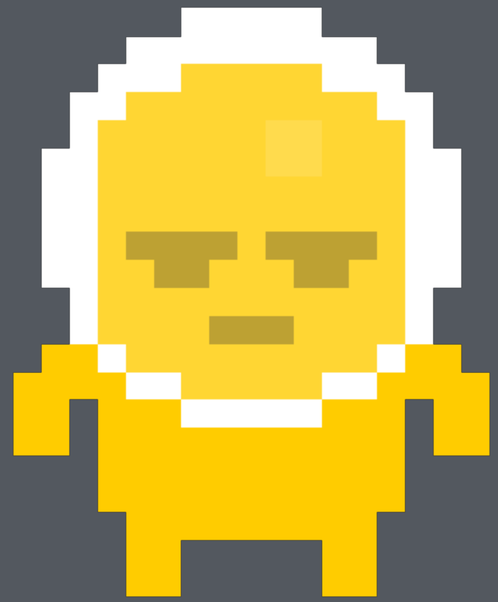
```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, surface->w, surface-
>h, 0, GL_RGBA, GL_UNSIGNED_BYTE, surface->pixels);
```

Texture **filtering**

Texture filtering parameters.

GL_LINEAR

Good for high resolution textures.



GL_NEAREST

Good for pixelart.

```
void glTexParameteri (GLenum target, GLenum pname,
GLint param);
```

Sets a **texture parameter** of the specified **texture target**.

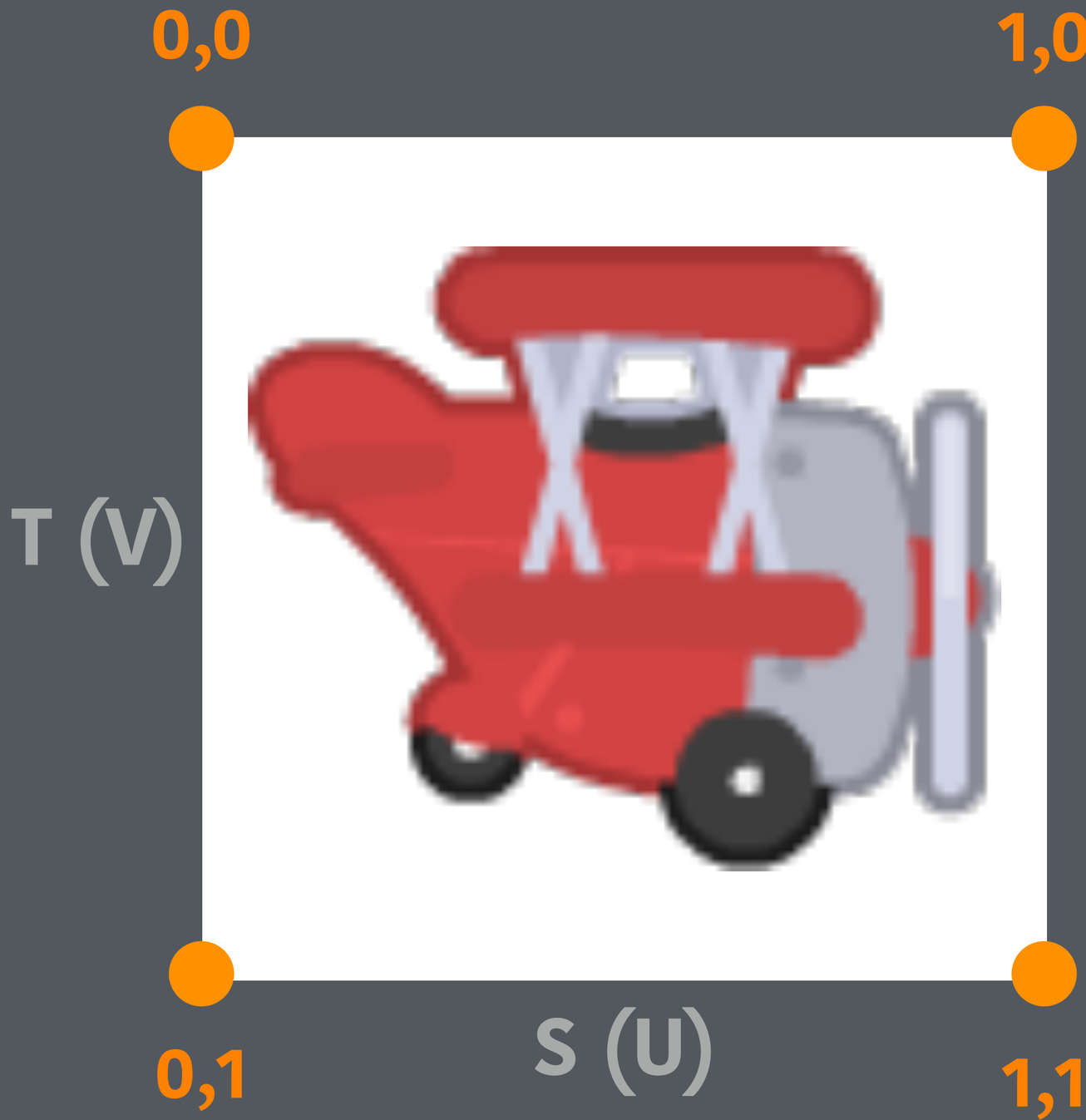
We **MUST** set the **texture filtering parameters** before the texture can be used.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

Putting it all together.

```
GLuint LoadTexture(const char *image_path) {
    SDL_Surface *surface = IMG_Load(image_path);

    GLuint textureID;
    glGenTextures(1, &textureID);
    glBindTexture(GL_TEXTURE_2D, textureID);

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, surface->w, surface->h, 0, GL_RGBA,
GL_UNSIGNED_BYTE, surface->pixels);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

    SDL_FreeSurface(surface);

    return textureID;
}
```

Texture **coordinates**.



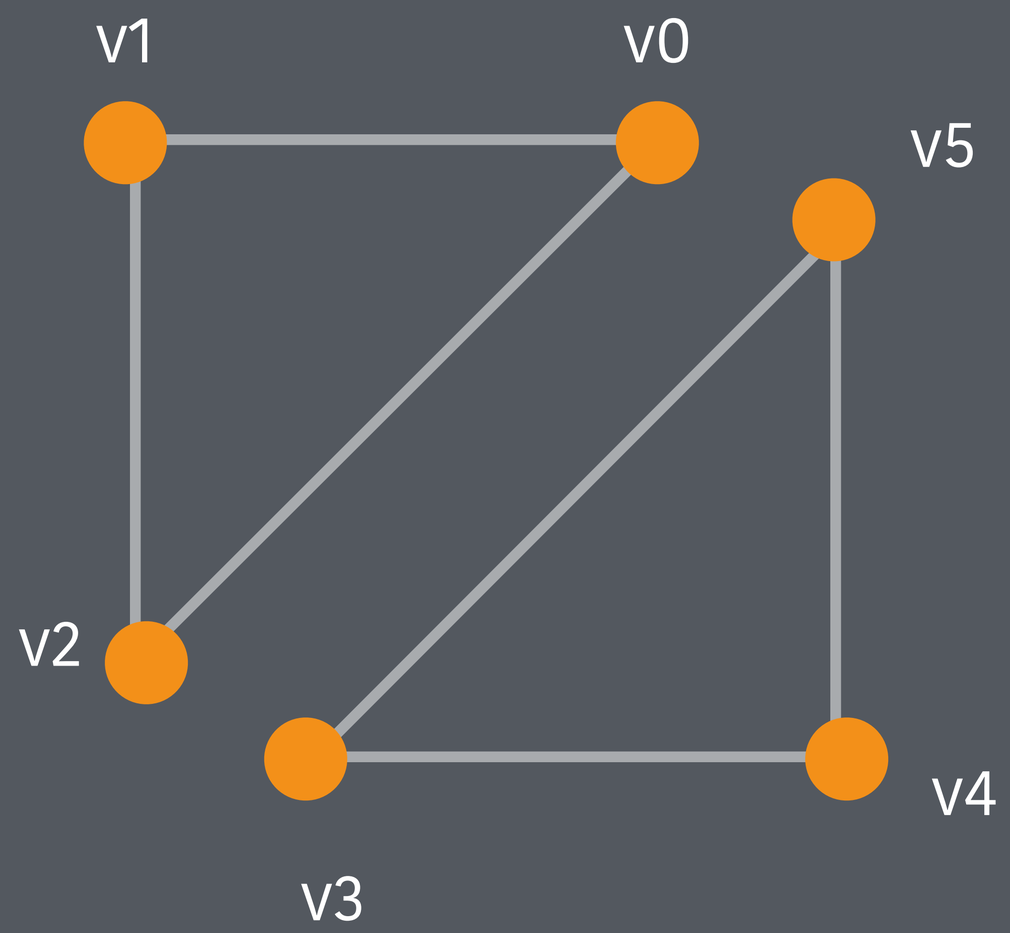
Texture coordinates are defined in 0-1 units called **UV coordinates**, not pixels!

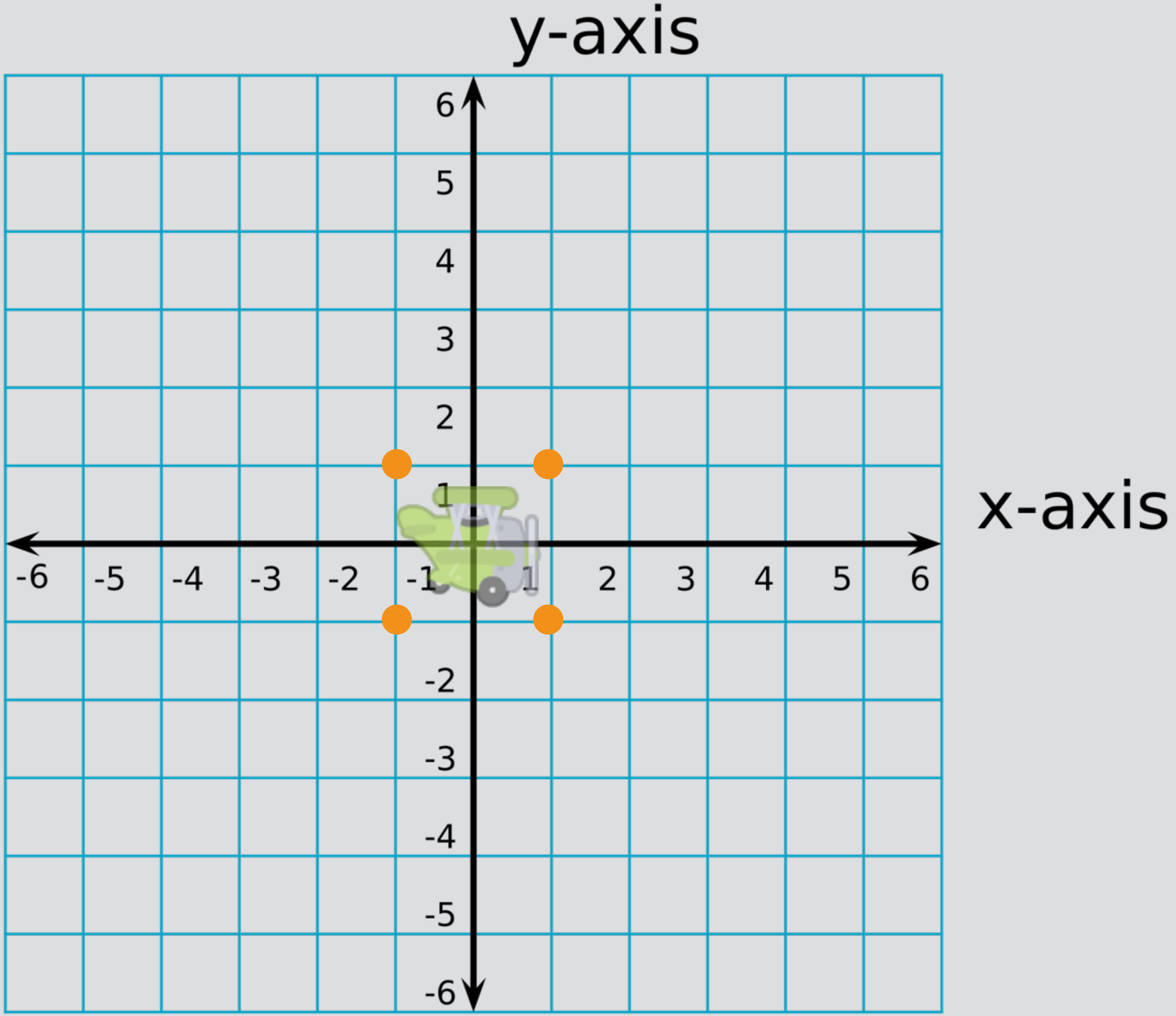
```
void glVertexAttribPointer (GLint index, GLint
size, GLenum type, GLboolean normalized, GLsizei
stride, const GLvoid *pointer);
```

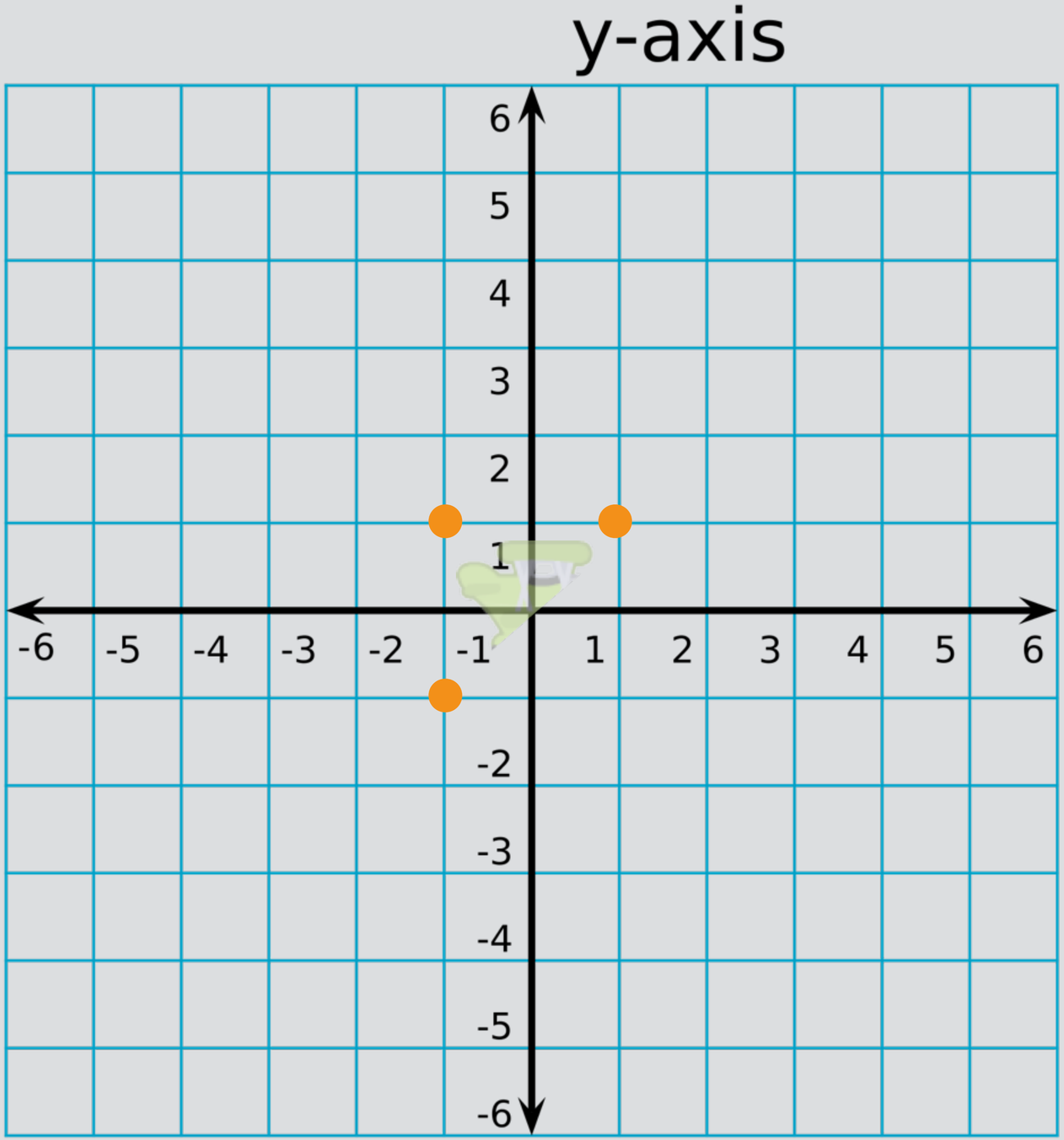
Defines an array of **vertex data**.

```
float texCoords[] = {0.0f, 1.0f, 0.0f, 0.0f, 1.0f, 0.0f};

glVertexAttribPointer(program.texCoordAttribute, 2, GL_FLOAT, false, 0, texCoords);
```

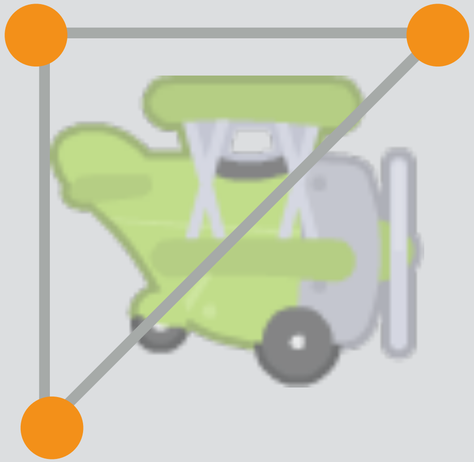
Drawing a **sprite**.

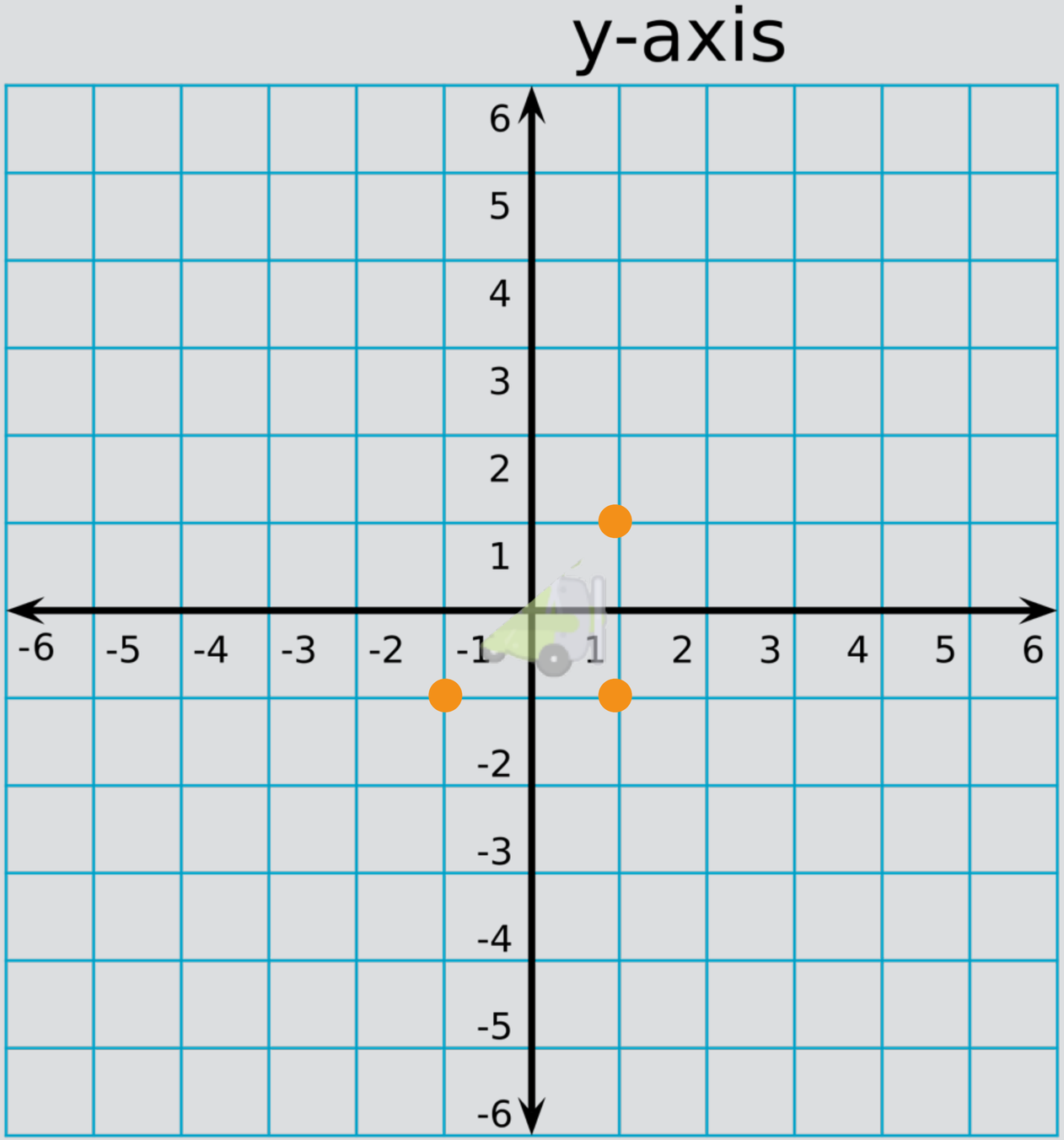

v1
v0
v5
v2
v4
v3


y-axis
x-axis
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6

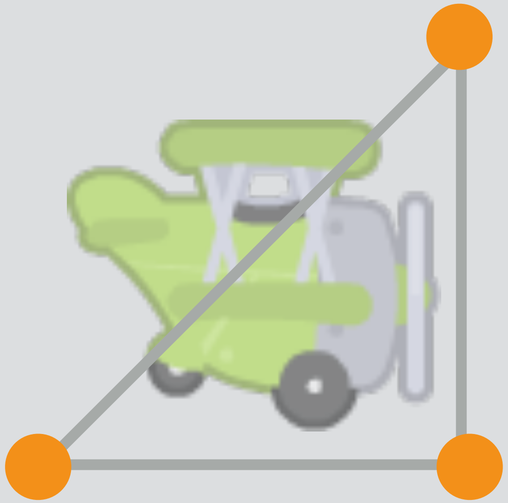


x-axis





x-axis



Drawing a **sprite**.

- Set **position attributes** for 2 triangles.
- Set **texture coordinate attributes** for 2 triangles.
- **Bind the texture** we want to use.
- **Draw** arrays.
- **Disable** attribute arrays.

Need to use a **shader program that supports textures!**

```
ShaderProgram program(RESOURCE_FOLDER”vertex_textured.glsl”, RESOURCE_FOLDER”fragment_textured.glsl");
```

Blending

Blending

Enabling blending

```
glEnable(GL_BLEND);
```

Common blending functions.

Alpha blending

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Additive blending

```
glBlendFunc (GL_SRC_ALPHA, GL_ONE);
```

Keeping **time**.

In **setup**

```
float lastFrameTicks = 0.0f;
```

In **game loop**

```
float ticks = (float)SDL_GetTicks()/1000.0f;
float elapsed = ticks - lastFrameTicks;
lastFrameTicks = ticks;
```

elapsed is how many seconds **elapsed since last frame**.

We will use this value to **move everything** in our game.

Basic time-based **animation**.

```
angle += elapsed;
DrawSprite(emojiTexture, 0.0, 0.0, angle);
```

Assignment #1

- Create a **simple 2D scene** using **textured** polygons.
- You can use **any images you want**, but feel free to use the assets in the class github repo.
- At least **one element** must be **animated**.
- You must use at least **3 different textures**.
- Commit the source to your **github repository** and email me the link.