Textures
Texture Atlas
Sprite Sheets
Fonts

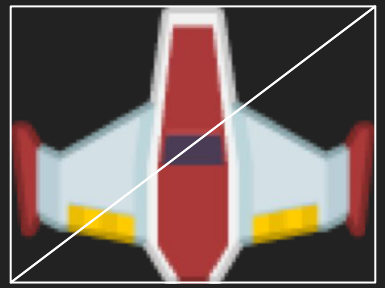
Textures
(Review)

Texture Coordinates

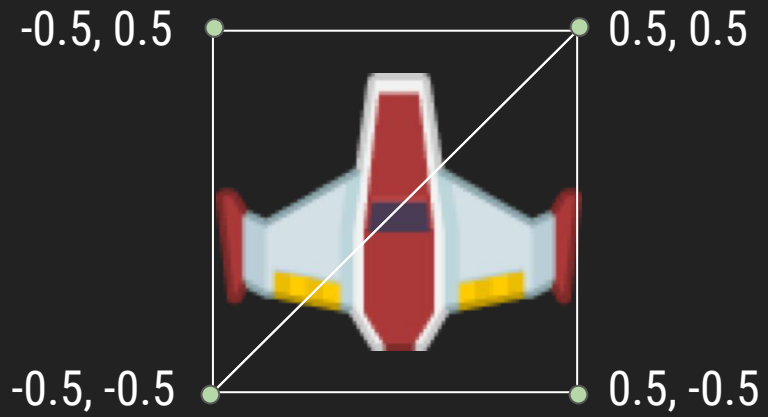
Texture coordinates are referred to as UV coordinates (X, Y and Z were already taken) :)

Notice the range from 0.0 to 1.0 and not by pixels.

2D Sprite Made of 2 Triangles

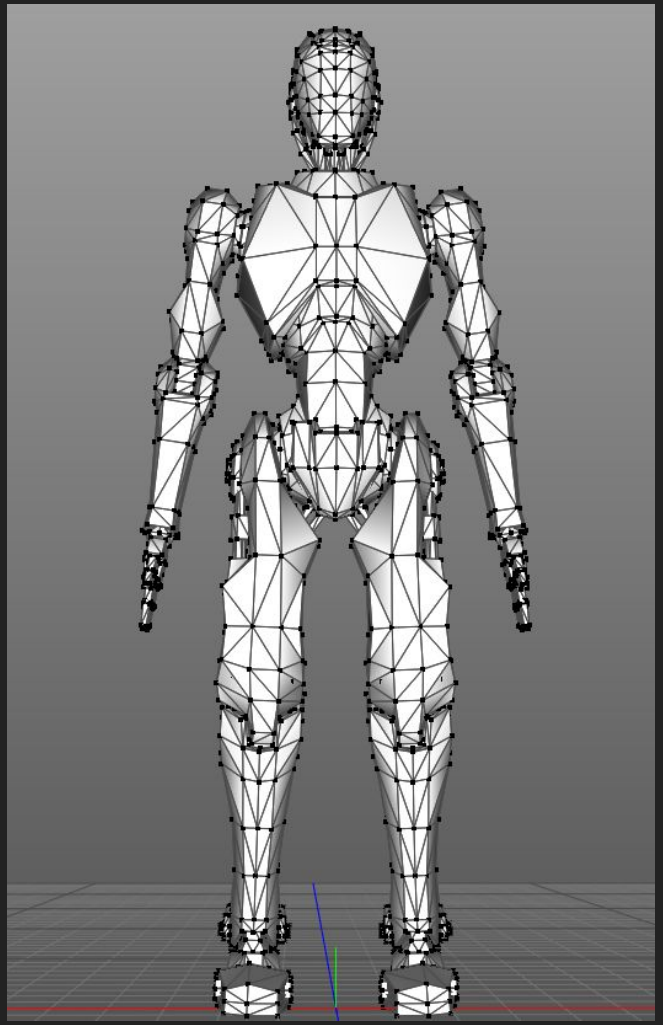


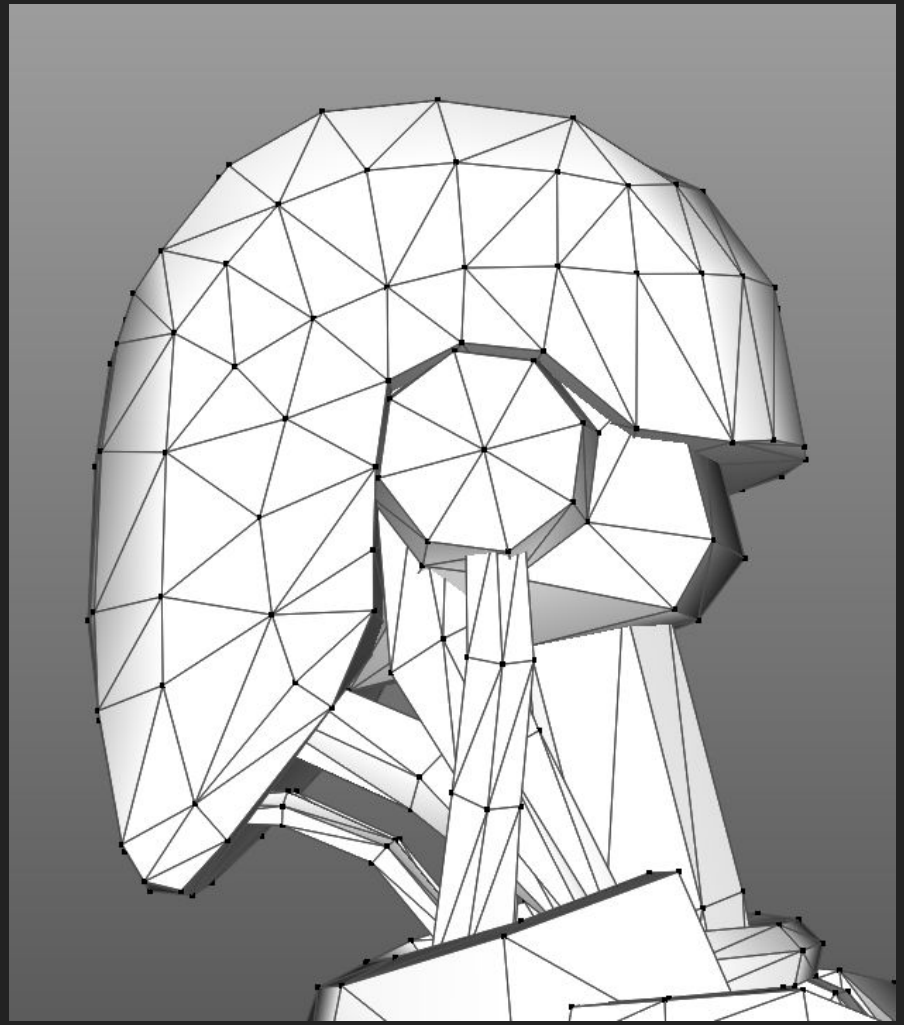
Need to match vertices to UV coordinates

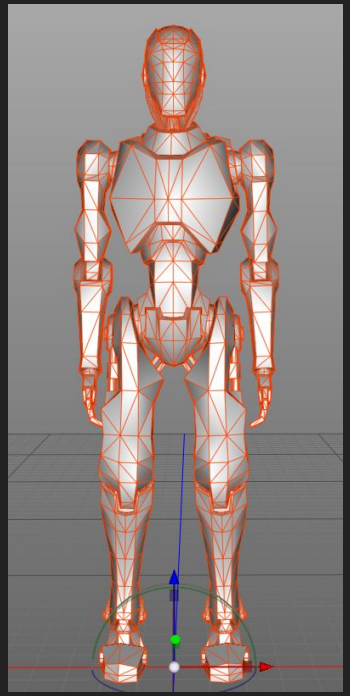


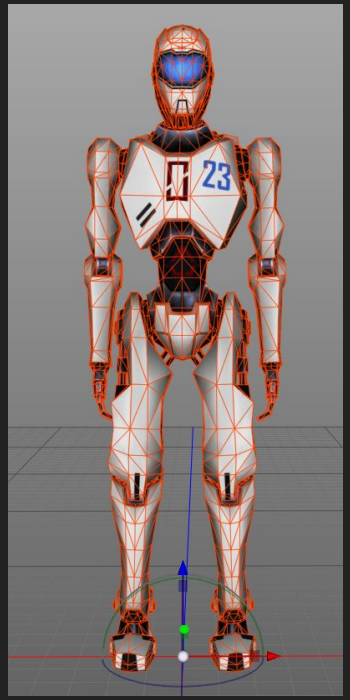

```
float vertices[]  = {-0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5};
float texCoords[] = {0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0};
```

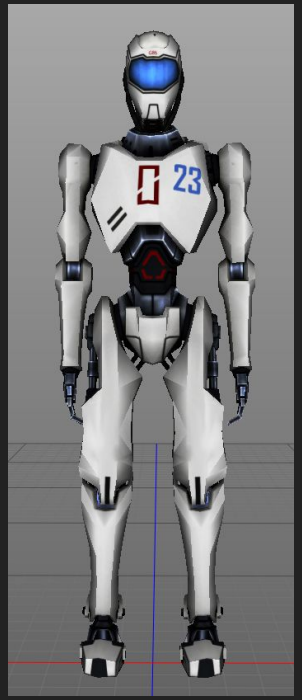
Portion of Texture





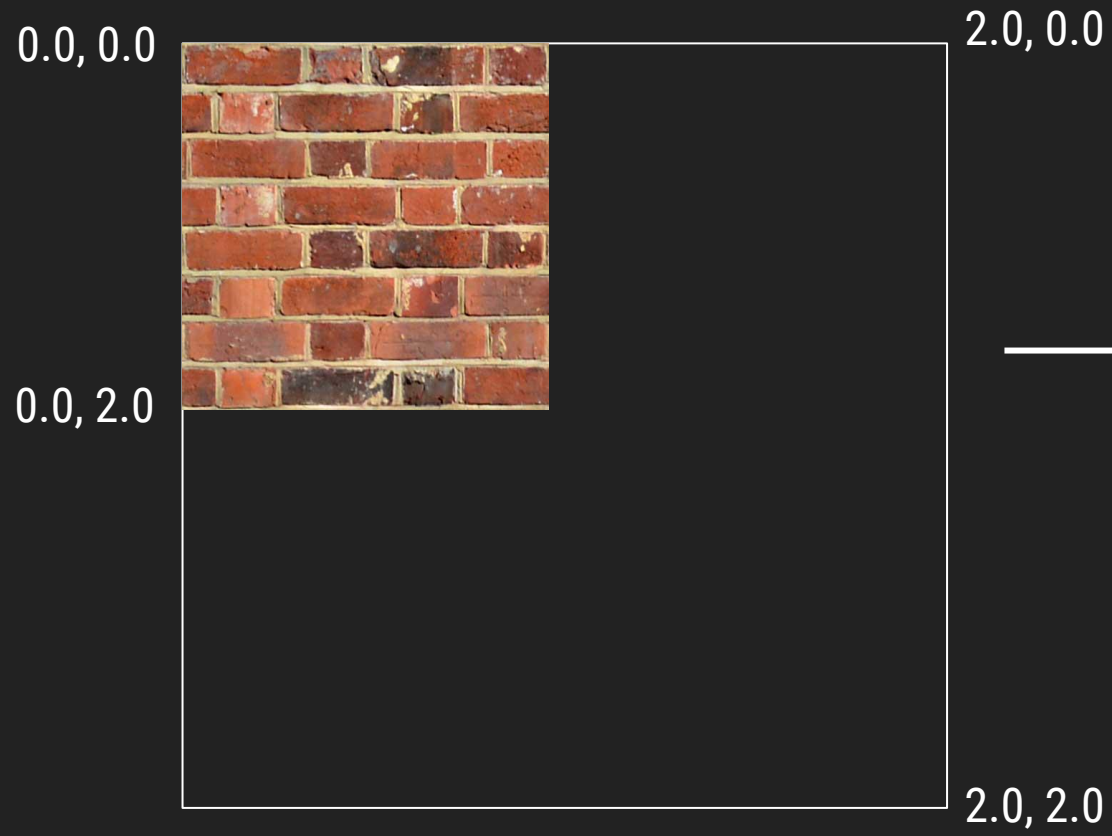






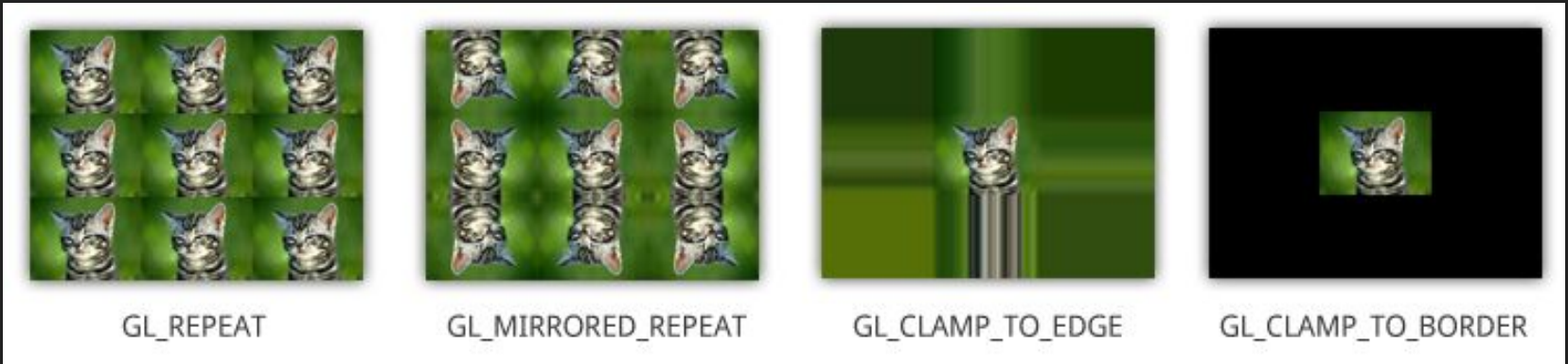
Texture Wrap Mode


```
float vertices[]  = { -0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5 };
float texCoords[] = { 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0 };
```




```
float vertices[]  = { -0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5 };
float texCoords[] = { 0.0, 2.0, 2.0, 2.0, 2.0, 0.0, 0.0, 2.0, 2.0, 0.0, 0.0, 0.0 };
```

Texture Wrap Mode



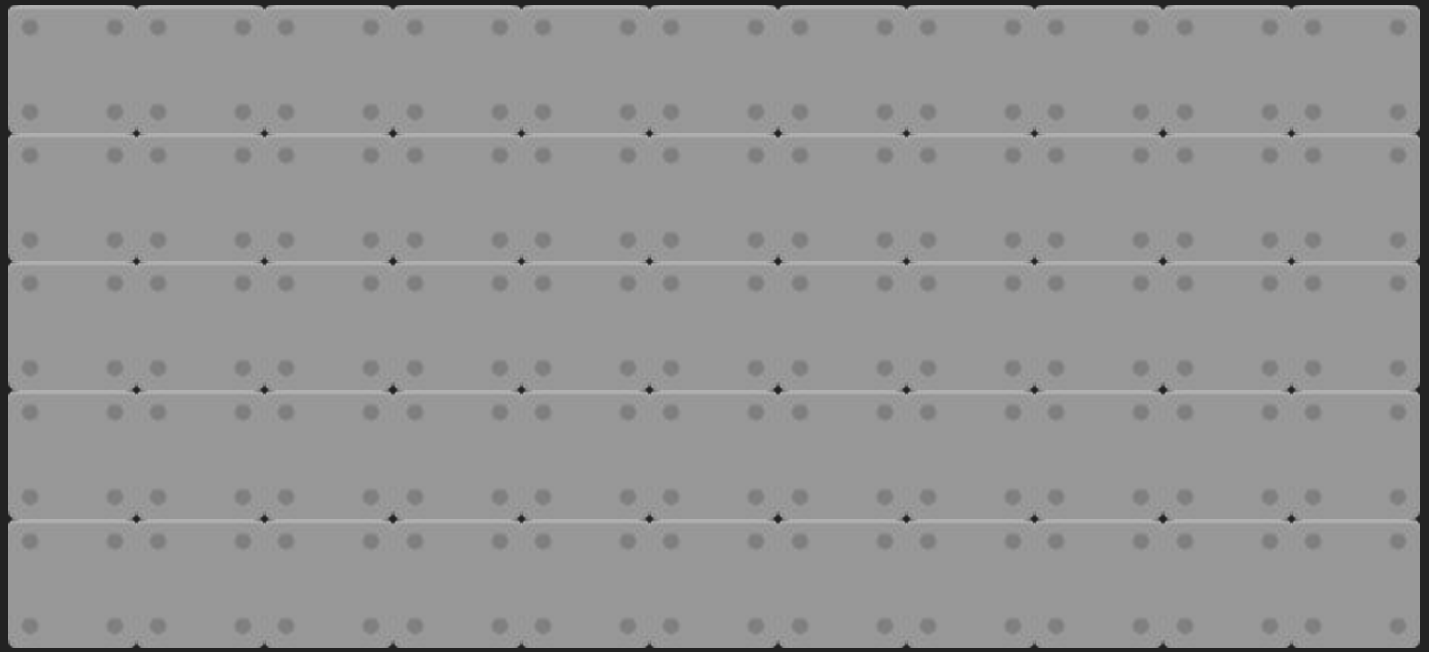
From https://open.gl/textures

Texture Wrap Mode

You can add these 2 lines.

```
GLuint LoadTexture(const char* filePath) {
    int w, h, n;
    unsigned char* image = stbi_load(filePath, &w, &h, &n, STBI_rgb_alpha);

    if (image == NULL) {
        std::cout << "Unable to load image. Make sure the path is correct\n";
        assert(false);
    }

    GLuint textureID;
    glGenTextures(1, &textureID);
    glBindTexture(GL_TEXTURE_2D, textureID);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, w, h, 0, GL_RGBA, GL_UNSIGNED_BYTE, image);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

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

    stbi_image_free(image);
    return textureID;
}
```

Repeating Tiles

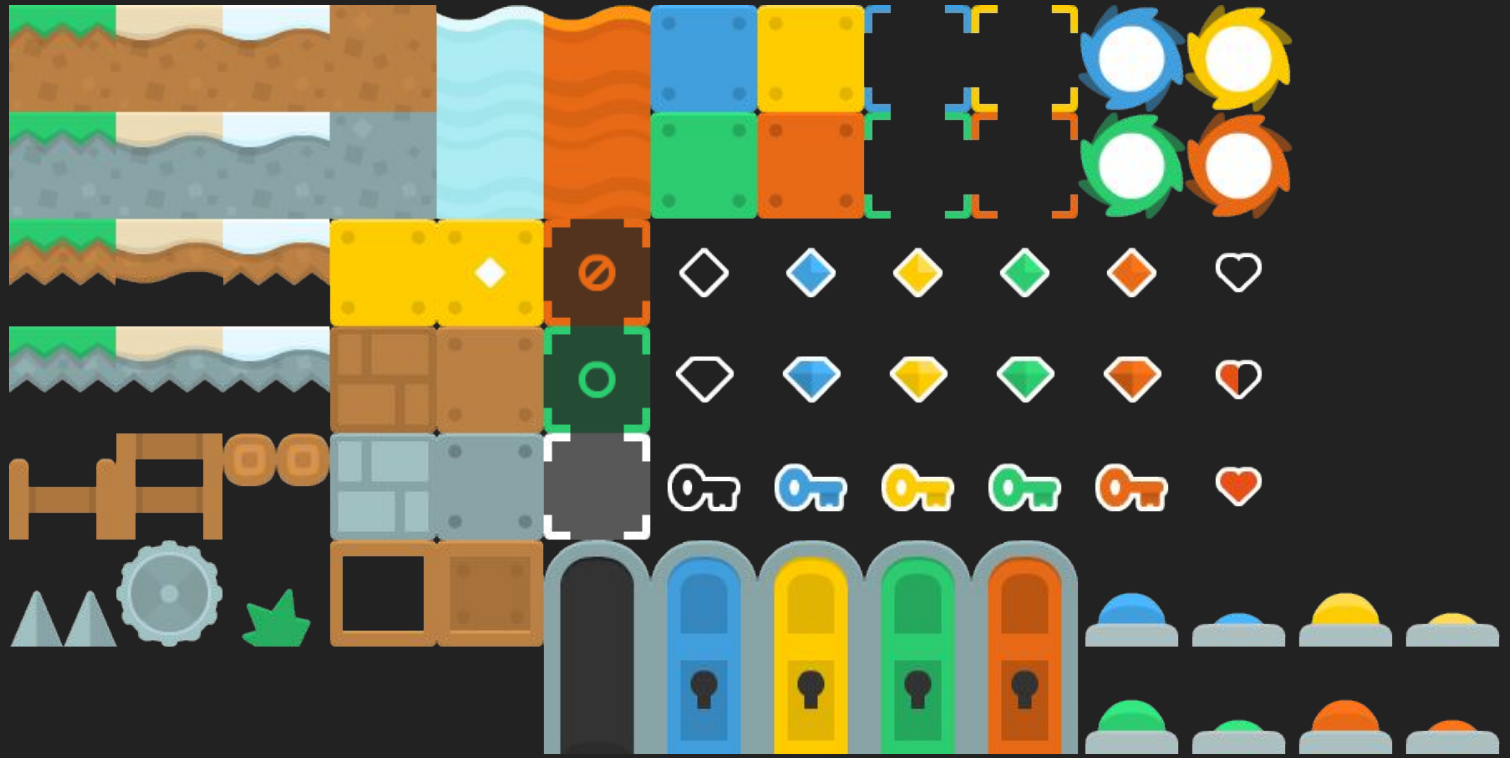


Texture Atlas

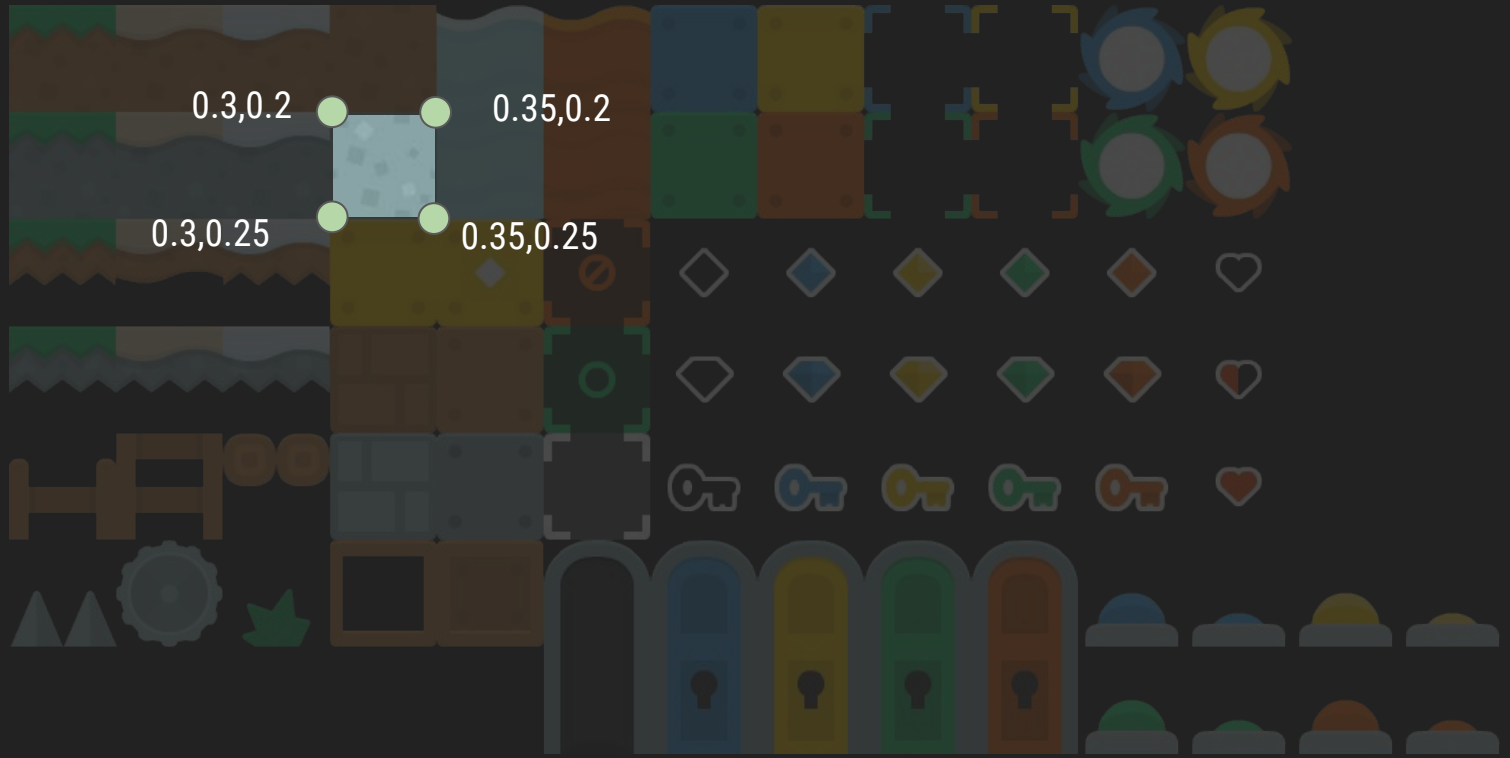
(Multiple Sprites in a Single Texture)

Sprite Sheet

Tileset



Tileset



3D Game Example

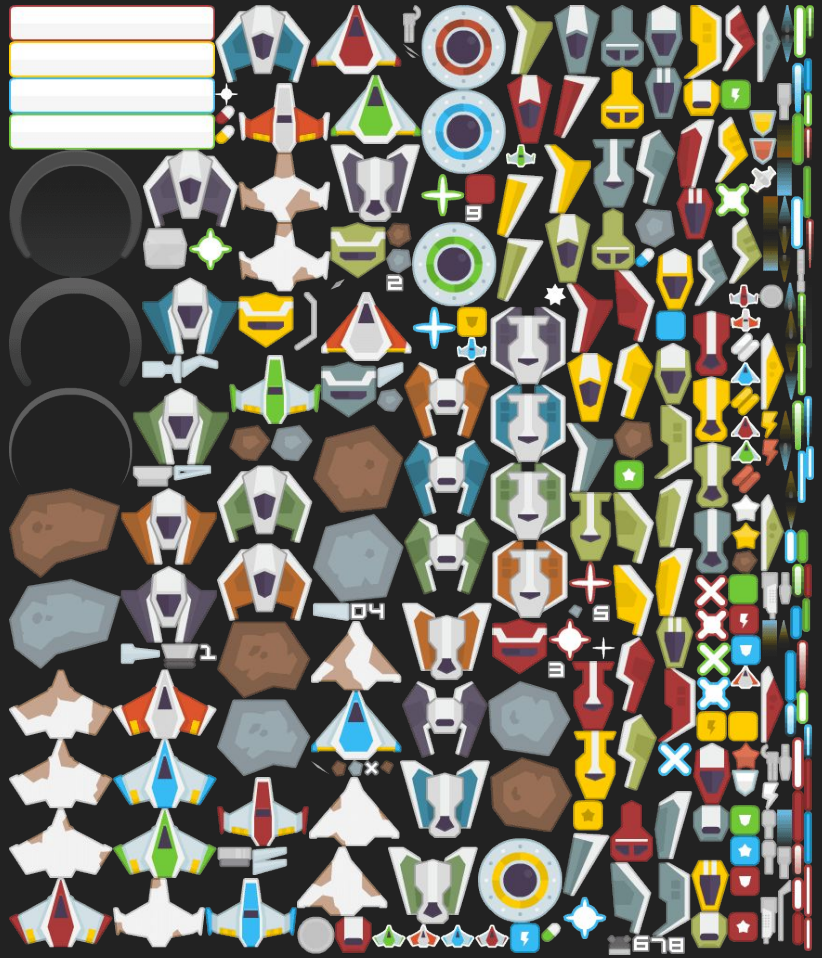
Fonts!


! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [\ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~
€ ‚ f „ … † ‡ ˆ ‰ Š ‹ Œ Ž
‘ ’ “ ” • – — ˜ ™ š › œ ž Ÿ
¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯
° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿
À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï
Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß
à á â ã ä å æ ç è é ê ë ì í î ï
ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

We are going to work with evenly spaced texture atlases.

Not Evenly Spaced

(you can not make a uniform grid on this)



Evenly Spaced

Evenly Spaced

Evenly Spaced

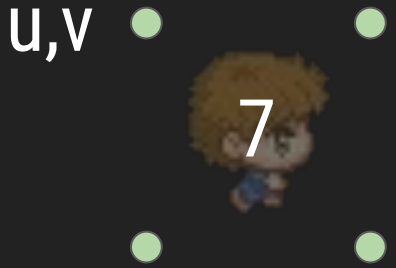
```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~
```

Drawing a Single Sprite

(From a Texture Atlas)

george_0.png

We need the UV coordinates of the individual sprite.

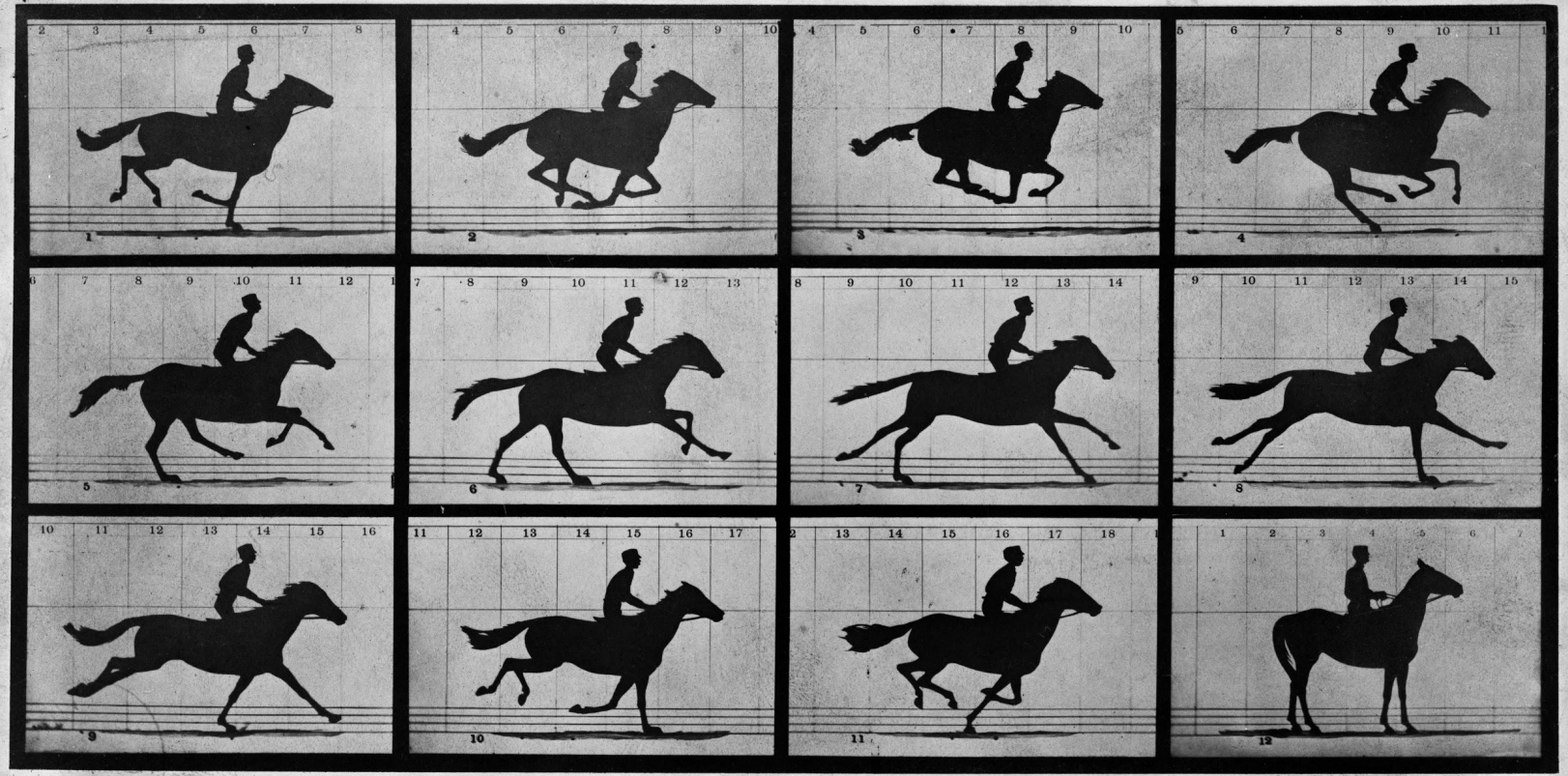



```
float u = (float)(index % cols) / (float)cols;
float v = (float)(index / cols) / (float)rows;

float width = 1.0f / (float)cols;
float height = 1.0f / (float)rows;

float texCoords[] = { u, v + height, u + width, v + height, u + width, v,
        u, v + height, u + width, v, u, v};

float vertices[]  = { -0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5 };
```

```
void DrawSpriteFromTextureAtlas(ShaderProgram *program, GLuint textureID, int index)
{
    float u = (float)(index % cols) / (float)cols;
    float v = (float)(index / cols) / (float)rows;

    float width = 1.0f / (float)cols;
    float height = 1.0f / (float)rows;

    float texCoords[] = { u, v + height, u + width, v + height, u + width, v,
        u, v + height, u + width, v, u, v};

    float vertices[]  = { -0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5 };

    glBindTexture(GL_TEXTURE_2D, textureID);

    glVertexAttribPointer(program->positionAttribute, 2, GL_FLOAT, false, 0, vertices);
    glEnableVertexAttribArray(program->positionAttribute);

    glVertexAttribPointer(program->texCoordAttribute, 2, GL_FLOAT, false, 0, texCoords);
    glEnableVertexAttribArray(program->texCoordAttribute);

    glDrawArrays(GL_TRIANGLES, 0, 6);

    glDisableVertexAttribArray(program->positionAttribute);
    glDisableVertexAttribArray(program->texCoordAttribute);
}
```

```
void Render() {
    glClear(GL_COLOR_BUFFER_BIT);

    program.SetModelMatrix(modelMatrix);
    DrawSpriteFromTextureAtlas(program, playerTextureID, 7);

    SDL_GL_SwapWindow(displayWindow);
}
```

Animation!



MORSE'S Gallery, 417 Montgomery St., San Francisco.

THE HORSE IN MOTION.

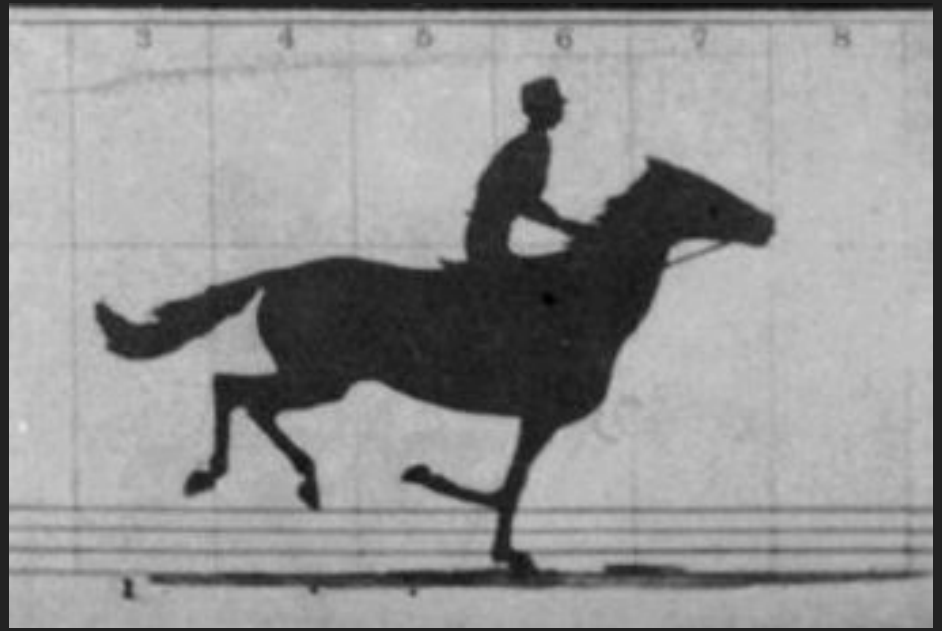
Illustrated by

MUYBRIDGE.

AUTOMATIC ELECTRO-PHOTOGRAPH.

"SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878.

The negatives of these photographs were made at intervals of twenty-seven inches of distance, and about the twenty-fifth part of a second of time; they illustrate consecutive positions assumed in each twenty-seven inches of progress during a single stride of the mare. The vertical lines were twenty-seven inches apart; the horizontal lines represent elevations of four inches each. The exposure of each negative was less than the two-thousandth part of a second.


3
4
5
6
7
8

Define indices of animation: (3, 7, 11, 15)

Have a timer.

Go to next frame when timer hits value.

If last frame (go to first) - looping.

```
// Inside of Initialization
playerTextureID = LoadTexture("george_0.png");
cols = 4;
rows = 4;
animIndices = new int[4] {3, 7, 11, 15};
animFrames = 4;
animIndex = 0;

// Inside of Update
animTime += deltaTime;

if (animTime >= 0.25f)
{
    animTime = 0.0f;
    animIndex++;
    if (animIndex >= animFrames)
    {
        animIndex = 0;
    }
}


// Inside of Render
DrawSpriteFromTextureAtlas(program, playerTextureID, animIndices[animIndex]);
```

Monospaced Font Rendering

For each character in a string

- Draw 2 Triangles
- Use UV coordinates for character

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Depending on the font texture, you may have to shift the character value.

Loop for each character in the string.

```
void DrawText(ShaderProgram *program, GLuint fontTextureID, std::string text,
                                  float size, float spacing, glm::vec3 position)
{
    float width = 1.0f / 16.0f;
    float height = 1.0f / 16.0f;

    std::vector<float> vertices;
    std::vector<float> texCoords;

    for(int i = 0; i < text.size(); i++) {

        int index = (int)text[i];
        float offset = (size + spacing) * i;

        float u = (float)(index % 16) / 16.0f;
        float v = (float)(index / 16) / 16.0f;
```

Setup the vertices and texCoords.

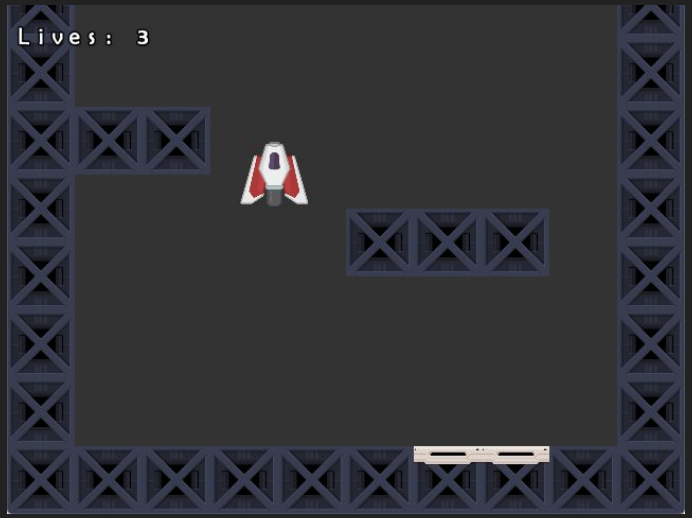
```
    vertices.insert(vertices.end(), {
        offset + (-0.5f * size), 0.5f * size,
        offset + (-0.5f * size), -0.5f * size,
        offset + (0.5f * size), 0.5f * size,
        offset + (0.5f * size), -0.5f * size,
        offset + (0.5f * size), 0.5f * size,
        offset + (-0.5f * size), -0.5f * size,
    });

    texCoords.insert(texCoords.end(), {
        u, v,
        u, v + height,
        u + width, v,
        u + width, v + height,
        u + width, v,
        u, v + height,
    });

} // end of for loop
```

After the vertices and texCoords are setup, we can draw using familiar code.

```
    glm::mat4 modelMatrix = glm::mat4(1.0f);
    modelMatrix = glm::translate(modelMatrix, position);
    program->SetModelMatrix(modelMatrix);

    glUseProgram(program->programID);

    glVertexAttribPointer(program->positionAttribute, 2, GL_FLOAT, false, 0, vertices.data());
    glEnableVertexAttribArray(program->positionAttribute);

    glVertexAttribPointer(program->texCoordAttribute, 2, GL_FLOAT, false, 0, texCoords.data());
    glEnableVertexAttribArray(program->texCoordAttribute);

    glBindTexture(GL_TEXTURE_2D, fontTextureID);
    glDrawArrays(GL_TRIANGLES, 0, (int)(text.size() * 6));

    glDisableVertexAttribArray(program->positionAttribute);
    glDisableVertexAttribArray(program->texCoordAttribute);
}
```

Example Usage

```
DrawText(&program, fontTextureID, "Lives: " + std::to_string(lives), 0.5f, -0.25f,
                                                                    glm::vec3(-4.75f, 3.3, 0));
```



Let’s Animate George!

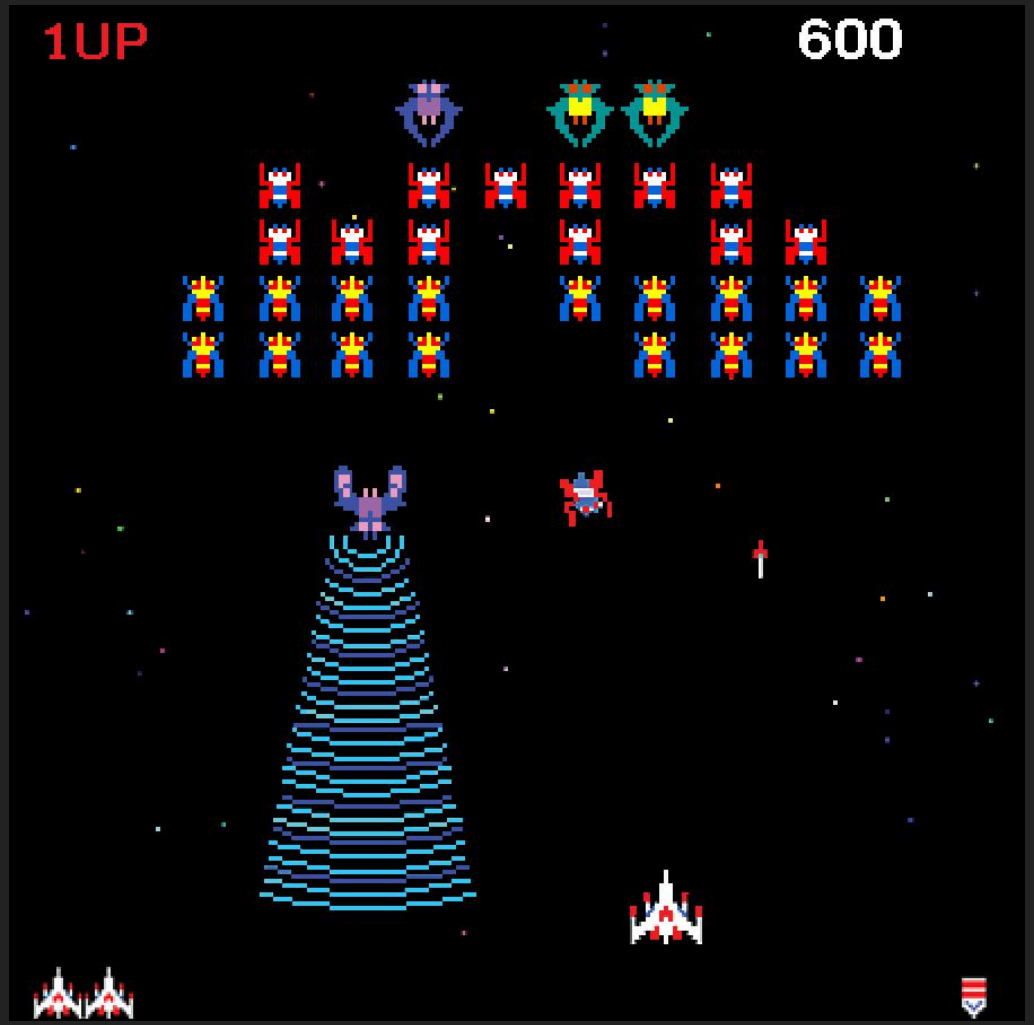
“I’ve got a bad feeling about this.”

Game Objects

Game State

Game Mode

Game Objects
(Entities)


1UP
600


250
100
20/20
Norad II
General Duke
Kills: 0
700/700
250/250
0
0
0
MENU

Entity.h

```
class Entity {
public:
    glm::vec3 position;
    glm::vec3 movement;
    float speed;

    GLuint textureID;

    glm::mat4 modelMatrix;

    Entity();

    void Update(float deltaTime);
    void Render(ShaderProgram *program);
};
```

Entity.cpp

```
#include "Entity.h"

Entity::Entity()
{
    position = glm::vec3(0);
    speed = 0;

    modelMatrix = glm::mat4(1.0f);
}

void Entity::Update(float deltaTime)
{
    position += movement * speed * deltaTime;

    modelMatrix = glm::mat4(1.0f);
    modelMatrix = glm::translate(modelMatrix, position);
}
```

Entity.cpp (continued)

```
void Entity::Render(ShaderProgram *program) {
    program->SetModelMatrix(modelMatrix);

    float vertices[]  = { -0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5 };
    float texCoords[] = { 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0 };

    glBindTexture(GL_TEXTURE_2D, textureID);

    glVertexAttribPointer(program->positionAttribute, 2, GL_FLOAT, false, 0, vertices);
    glEnableVertexAttribArray(program->positionAttribute);

    glVertexAttribPointer(program->texCoordAttribute, 2, GL_FLOAT, false, 0, texCoords);
    glEnableVertexAttribArray(program->texCoordAttribute);

    glDrawArrays(GL_TRIANGLES, 0, 6);

    glDisableVertexAttribArray(program->positionAttribute);
    glDisableVertexAttribArray(program->texCoordAttribute);
}
```

Object Pool

(optimizing for tons of objects)

Object Pool

Maximum number of objects.

Create/Allocate objects ahead of time.

Use a bool for active or not.

You can test with max amount of objects.

Less prone to memory leaks.

Object Pool

```
#define MAX_BULLETS 100;

int nextBullet = 0;
Entity bullets[MAX_BULLETS];

void initialize() {
    for (int i = 0; i < MAX_BULLETS; i++) {
        bullets[i].active = false;
    }
}

void fire() {
    bullet[nextBullet].position = // somewhere
    bullet[nextBullet].active = true;

    nextBullet++;
    if (nextBullet == MAX_BULLETS) nextBullet = 0;
}
```

Game State

entities, score, lives left, time left, etc.

Game State

```
struct GameState {
    Entity player;
    Entity enemies[10];
    Entity items[5];
    int score;
};

GameState state;
```

Game Mode

Game Mode

Arcade games typically feature an “attract” mode.
Also called “demo” mode. Sometimes seen in NES games.

Game plays itself, shows high scores, cut scenes, etc.

Rygar:
https://www.youtube.com/watch?v=jV2iT9LnCD4

Street Fighter II:
https://www.youtube.com/watch?v=TU1C1ihW2mg

Game Mode

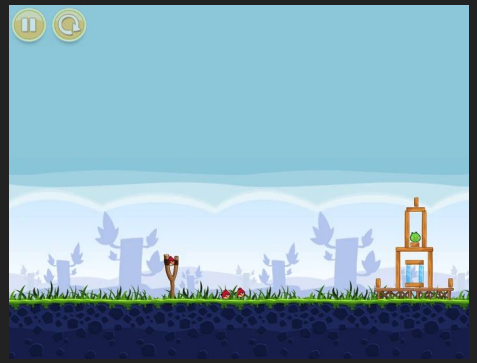
Modern Console/PC Games typically do not have Attract/Demo modes.

Main Menu

Chapter Select

Level Select

Cut Scene



Game Level

Win Screen

Game Mode

```
enum GameMode { MAIN_MENU, GAME_LEVEL, GAME_OVER };
GameMode mode = MAIN_MENU;
```

Game Mode

```
void ProcessInput() {
    switch (mode) {
        case MAIN_MENU:
            ProcessInputMainMenu();
            break;

        case GAME_LEVEL:
            ProcessInputGameLevel();
            break;

        case GAME_OVER:
            ProcessInputGameOver();
            break;
    }
}
```

Game Mode

```
void Update() {
    float ticks = (float)SDL_GetTicks() / 1000.0f;
    float deltaTime = ticks - lastTicks;
    lastTicks = ticks;

    switch (mode) {
        case MAIN_MENU:
            UpdateMainMenu(deltaTime);
            break;

        case GAME_LEVEL:
            UpdateGameLevel(deltaTime);
            break;

        case GAME_OVER:
            UpdateGameOver(deltaTime);
            break;
    }
}
```

Game Mode

```
void Render() {
    glClear(GL_COLOR_BUFFER_BIT);

   switch (mode) {
       case MAIN_MENU:
           RenderMainMenu();
           break;

       case GAME_LEVEL:
           RenderGameLevel();
           break;

       case GAME_OVER:
           RenderGameOver();
           break;
   }

    SDL_GL_SwapWindow(displayWindow);
}
```

Game Mode

(or make a class for each mode)

```
MainMenu mainMenu;
GameLevel gameLevel;
GameOver gameOver;

void Render() {
     switch (mode) {
          case GAME_LEVEL:
               gameLevel.Render();
          break;

          // .. other modes
     }
}
```

```
class GameLevel {
     GameState state;

     public void Render() {
          state.player.Render();
          for (int i = 0; i < enemies.length; i++) {
               state.enemies[i].Render();
          }
     }

     // Other stuff

}
```

Let’s Code!

Entity.h

Entity.cpp

Update main.cpp