Textures

Texture Atlas

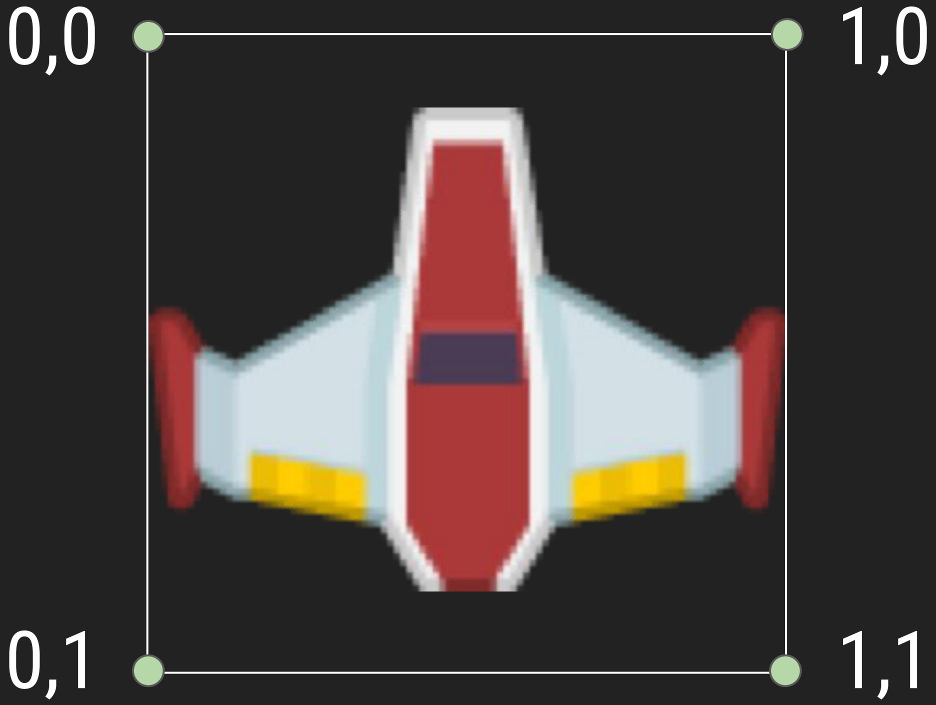
Sprite Sheets

Fonts

Textures

(Review)

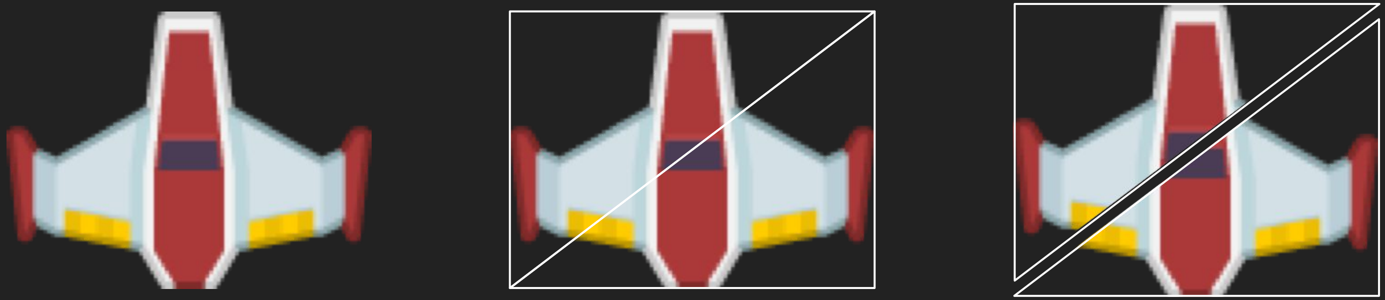
Texture Coordinates



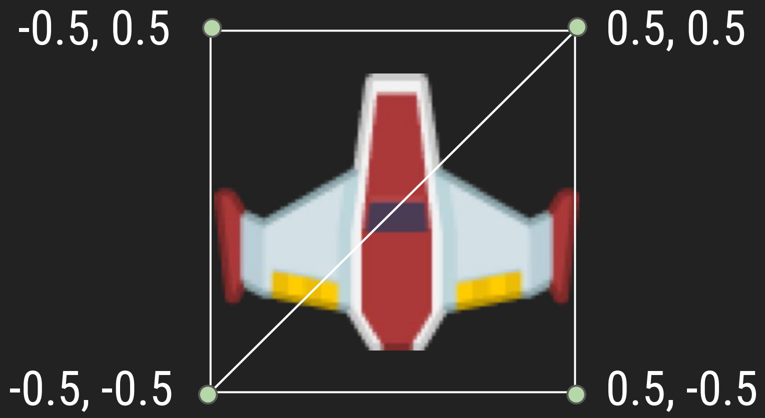
Texture coordinates are referred to as **UV** coordinates (X, Y and Z were already taken) :)

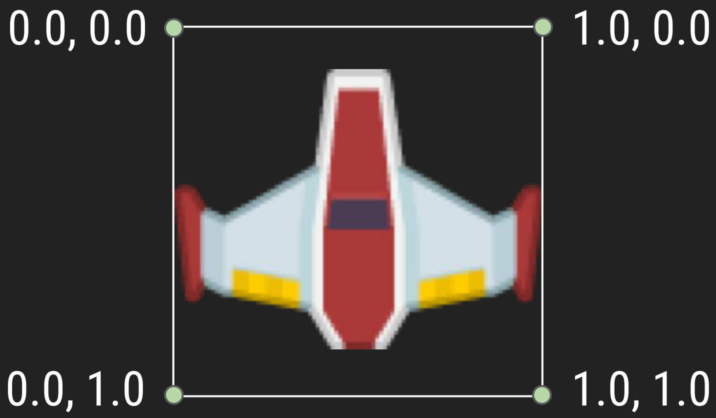
Notice the range from **0.0 to 1.0** and not by pixels.

2D Sprite Made of 2 Triangles



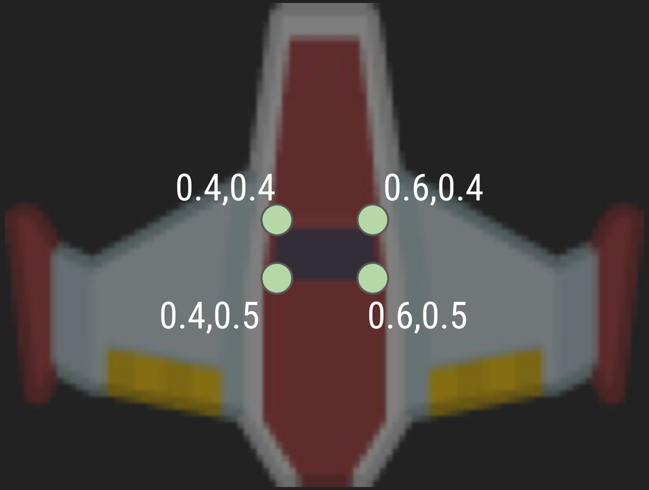
Need to match vertices to UV coordinates

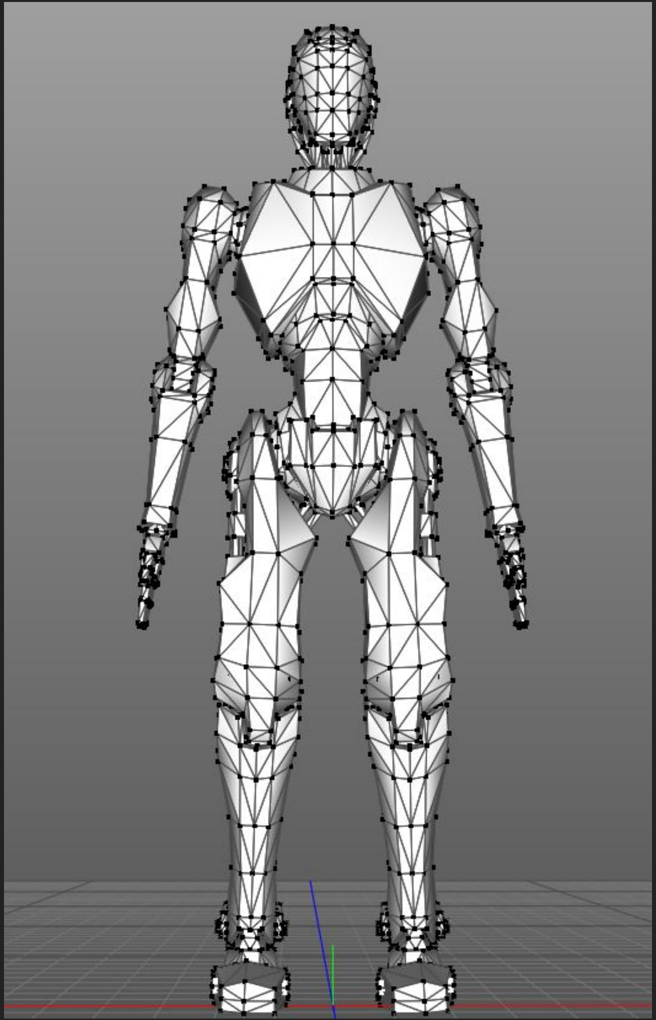
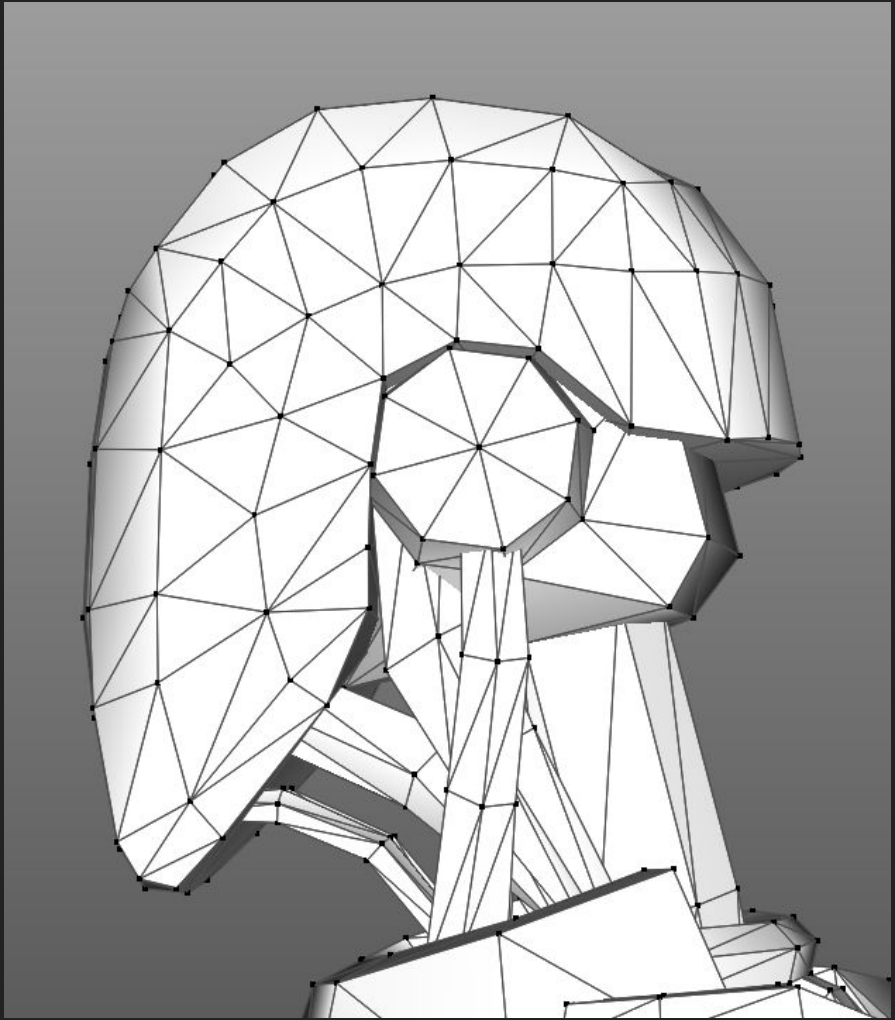


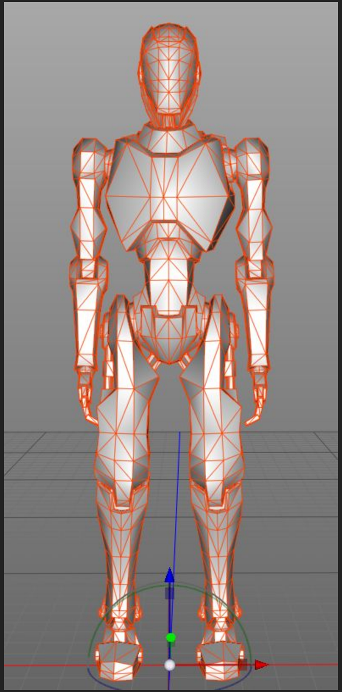
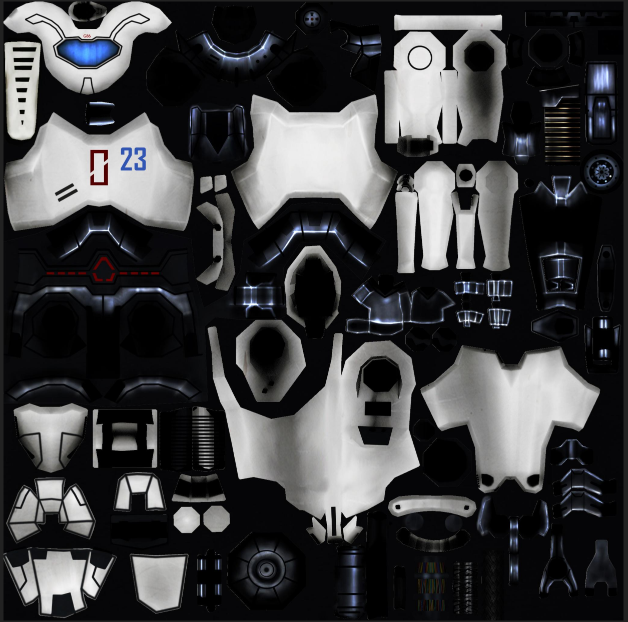
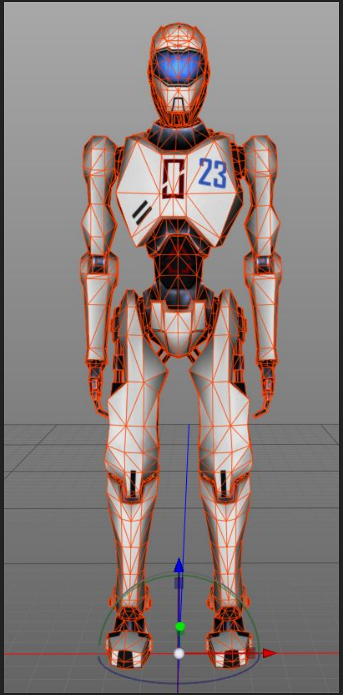
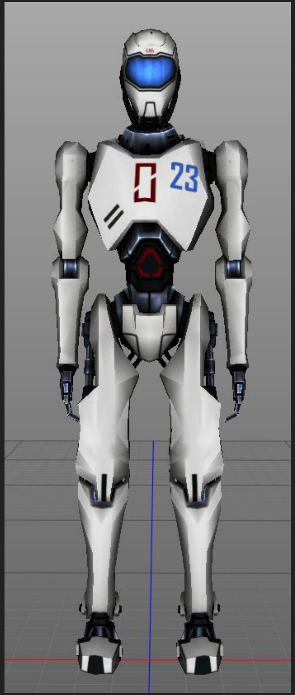


```
float vertices[]  = {-0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5};
float texCoords[] = {0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0};
```

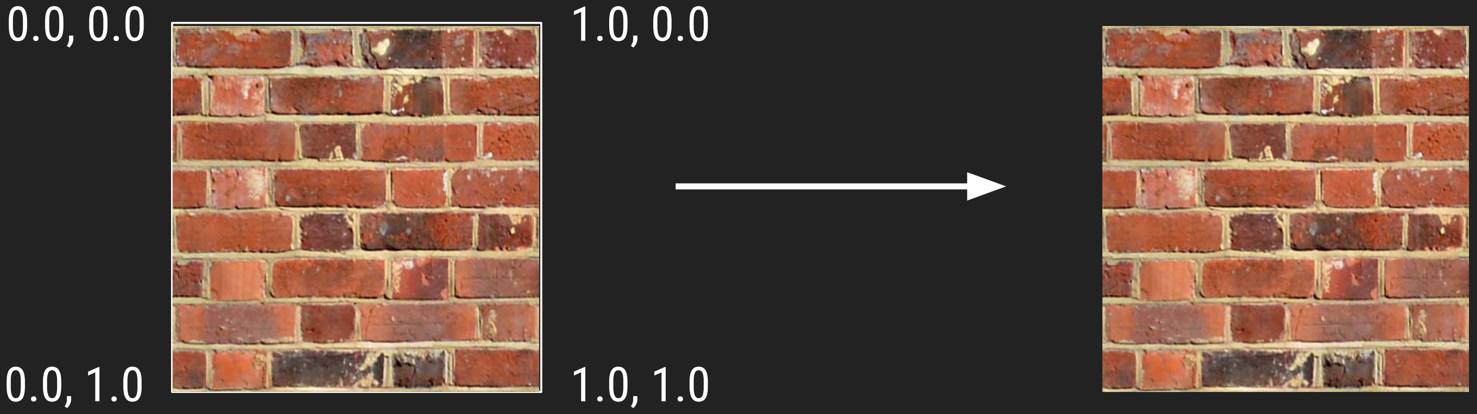
Portion of Texture



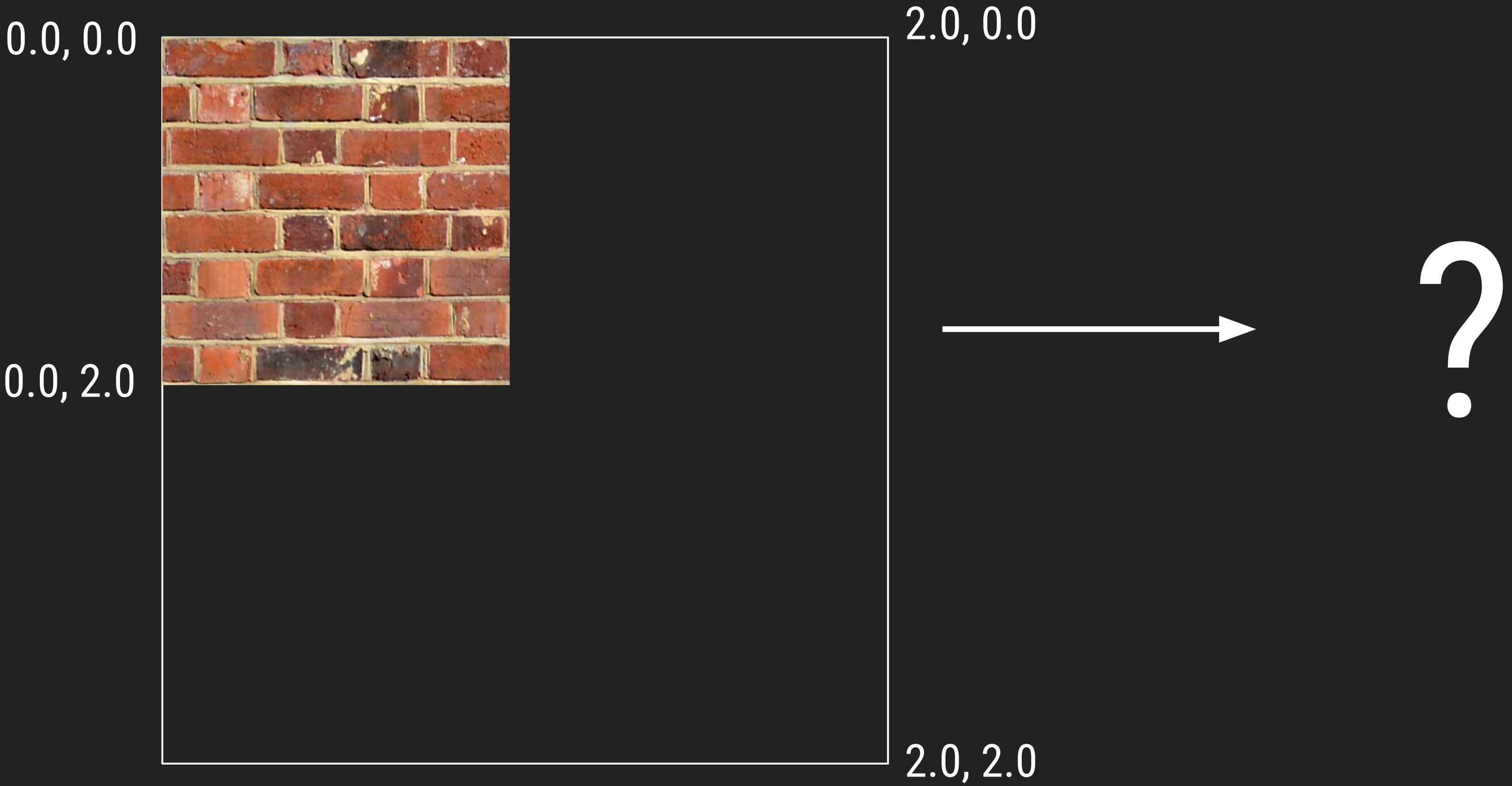



Texture Wrap Mode

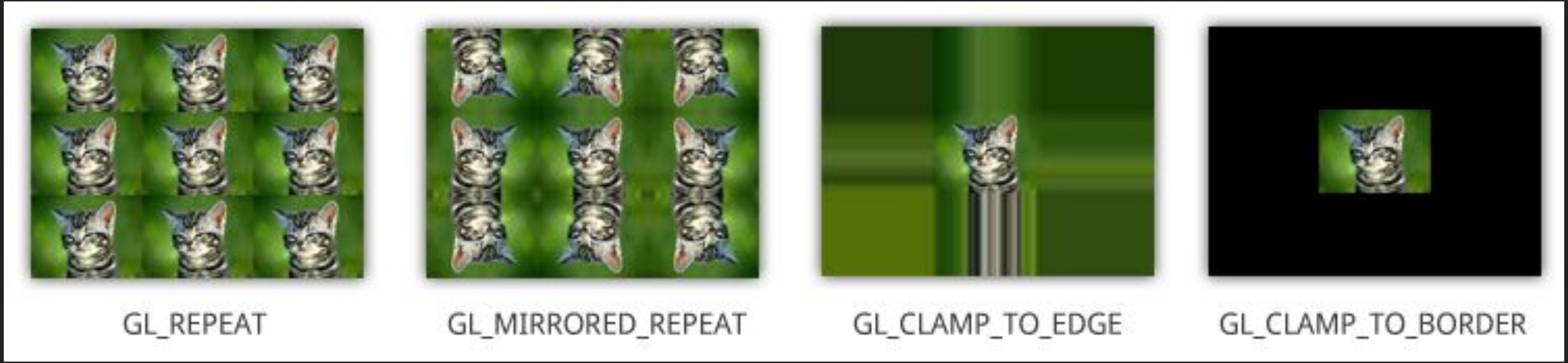


```
float vertices[]  = { -0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5 };
float texCoords[] = { 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0 };
```



```
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 };
```

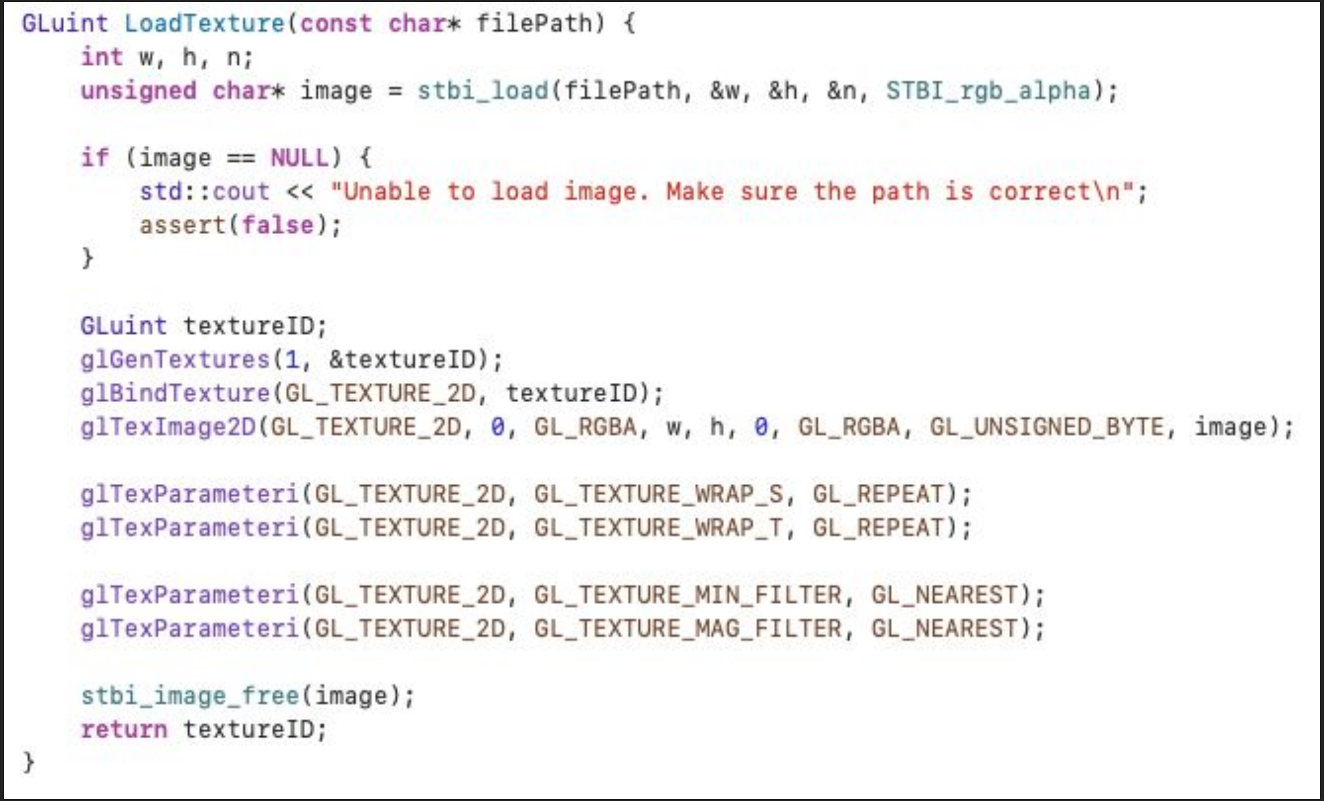
Texture Wrap Mode



From https://open.gl/textures

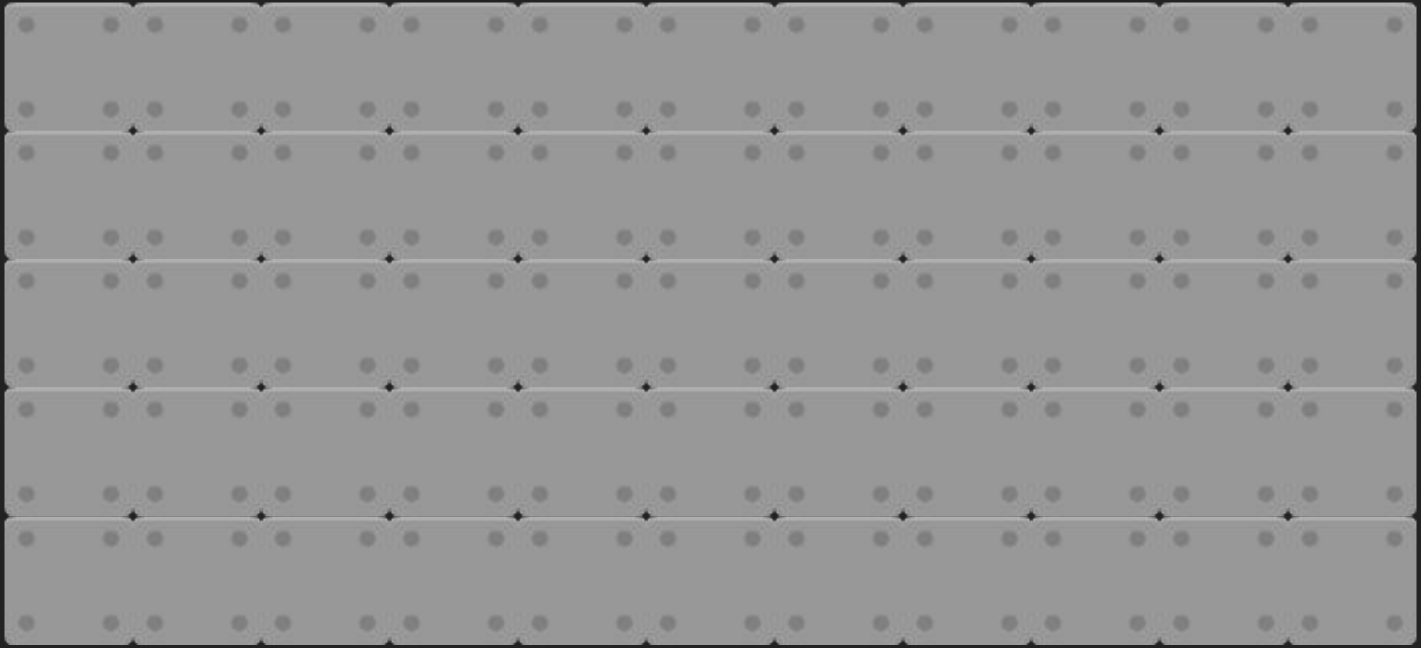
Texture Wrap Mode

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;
}
```

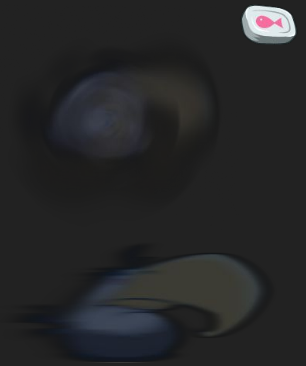
Repeating Tiles



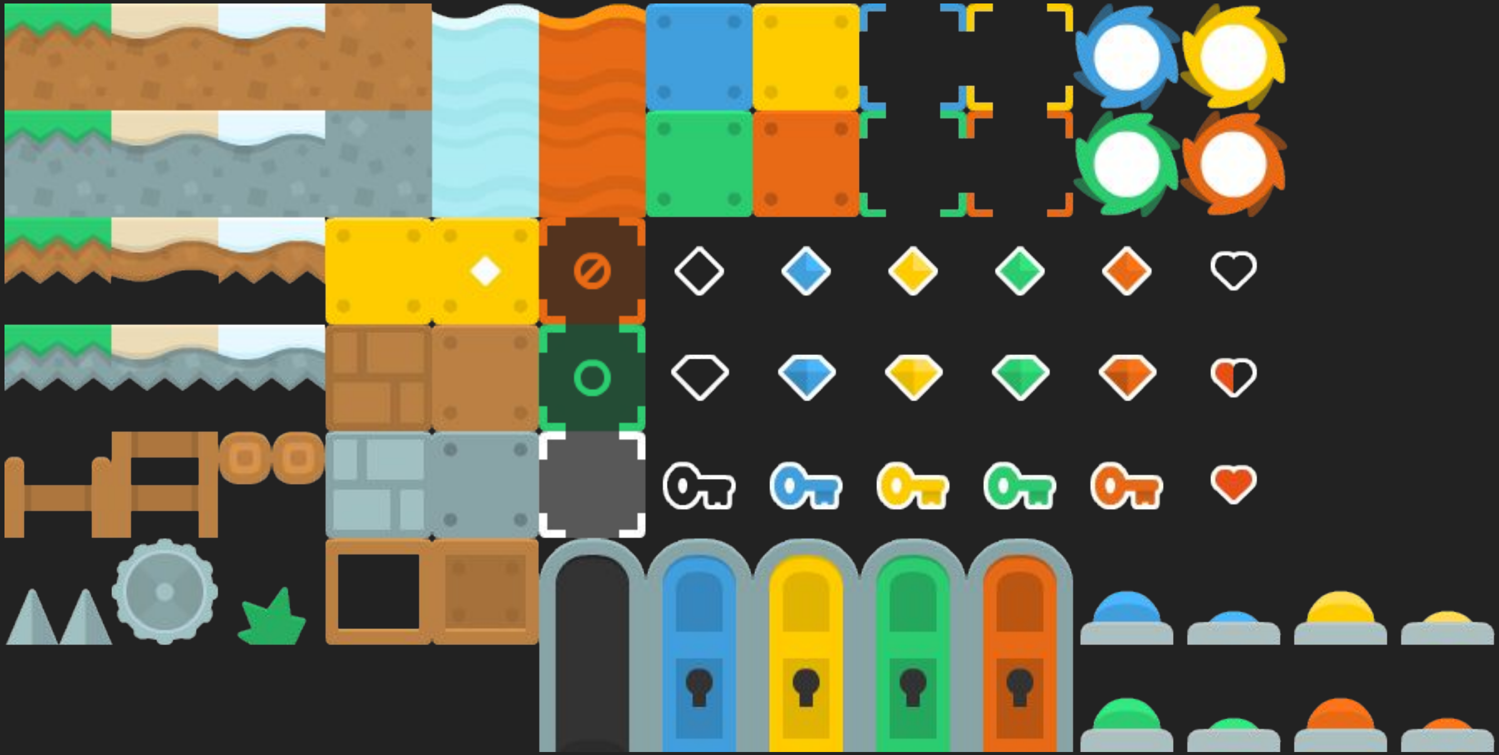
Texture Atlases

(Multiple Sprites in a Single Texture)

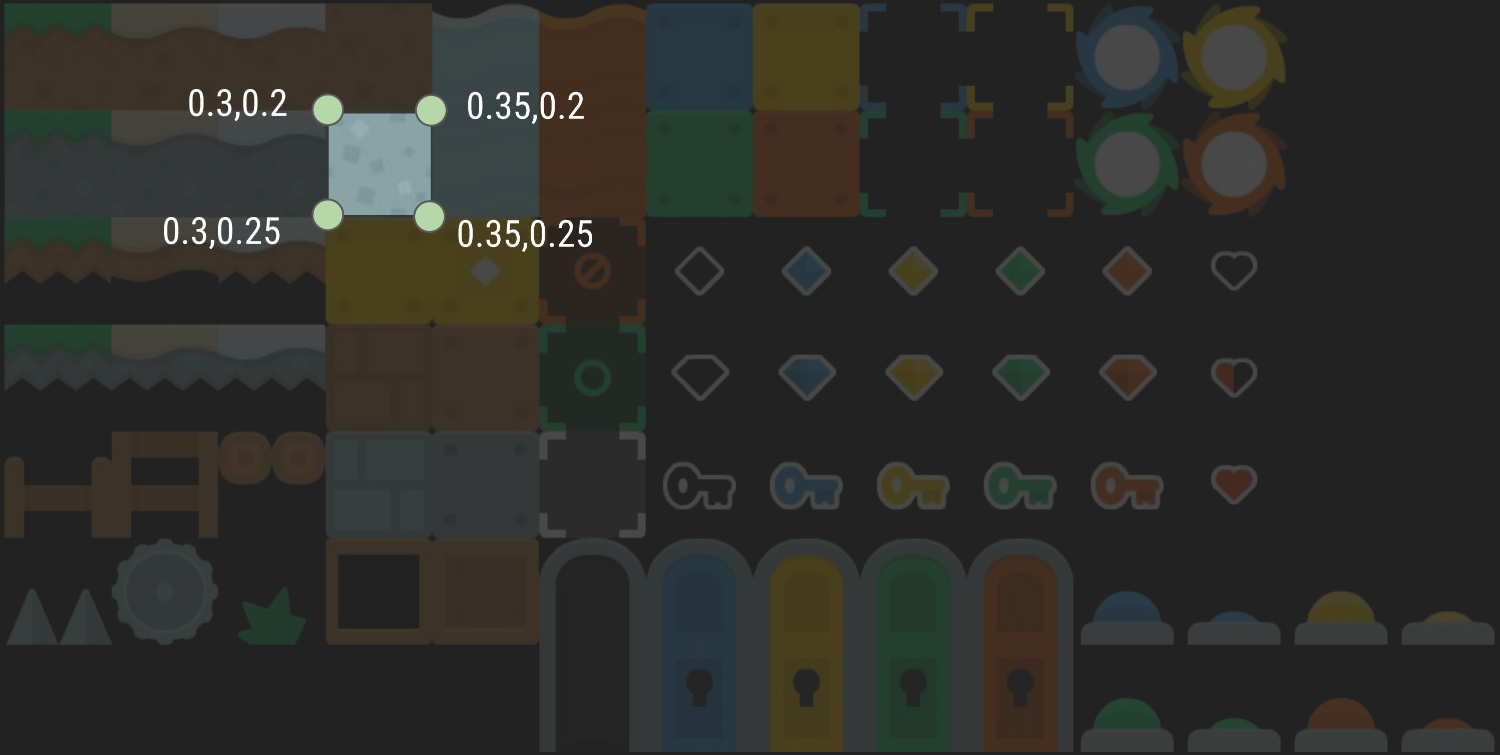
Sprite Sheet



Tileset



Tileset



3D Game Example

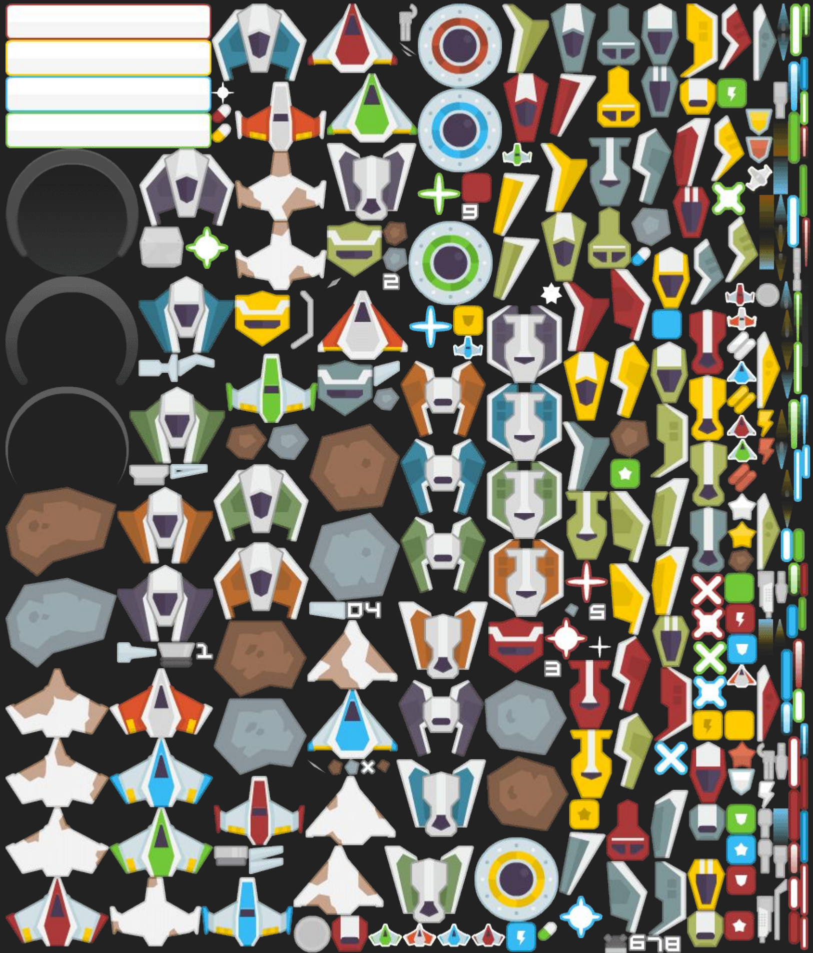
Fonts!

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
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	{	|	}	~	▯
€	▯	‚	ƒ	„	…	†	‡	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
Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	œ	ç	è	é	ê	ë	ì	í	î	ï
đ	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

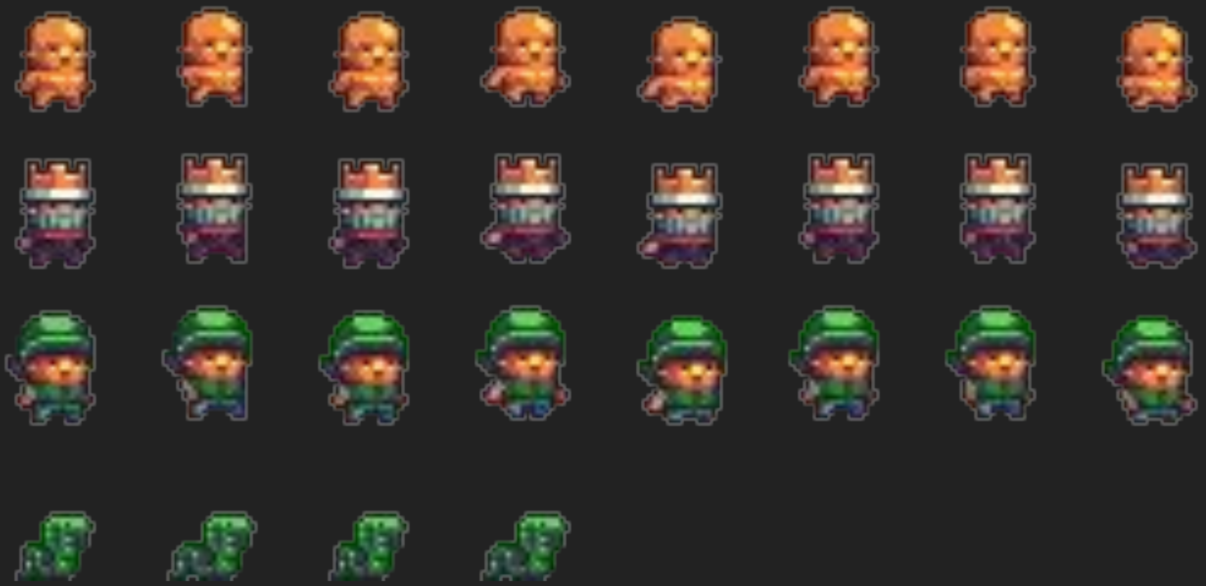
We are going to work with evenly spaced texture atlases.

Not Evenly Spaced

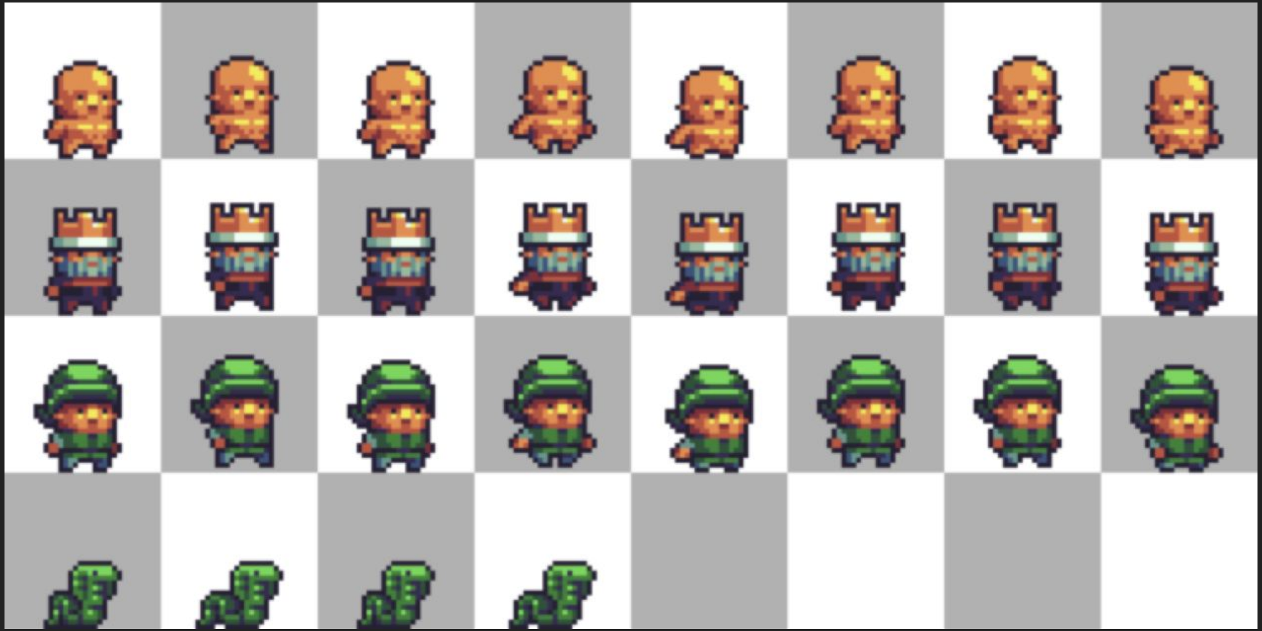
(you can not make a uniform grid on this)



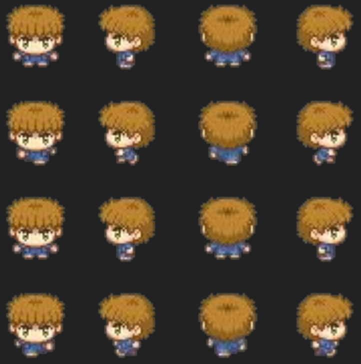
Evenly Spaced

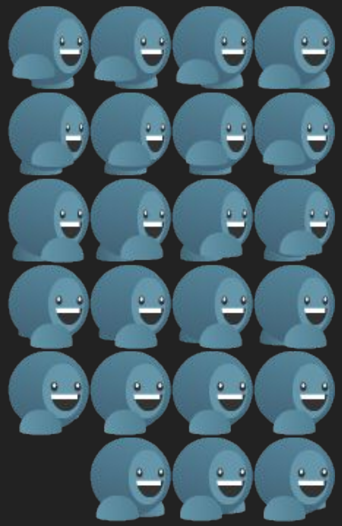


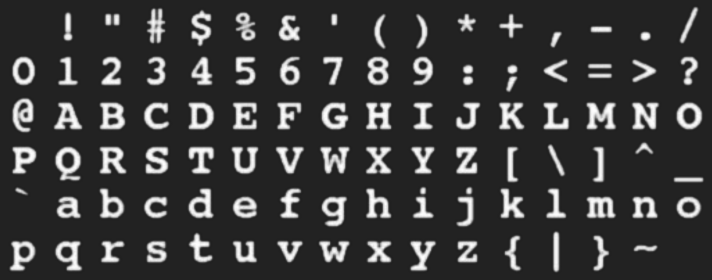
Evenly Spaced



Evenly Spaced



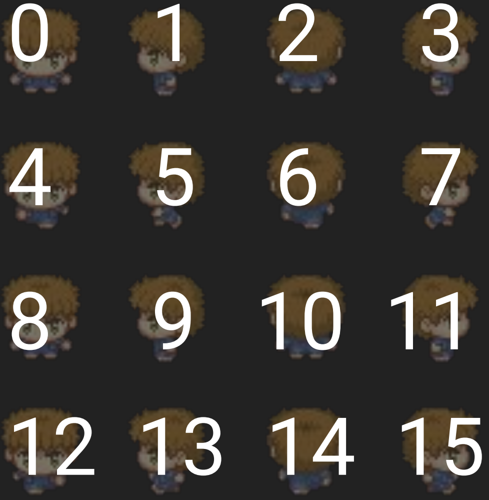



! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [\ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~

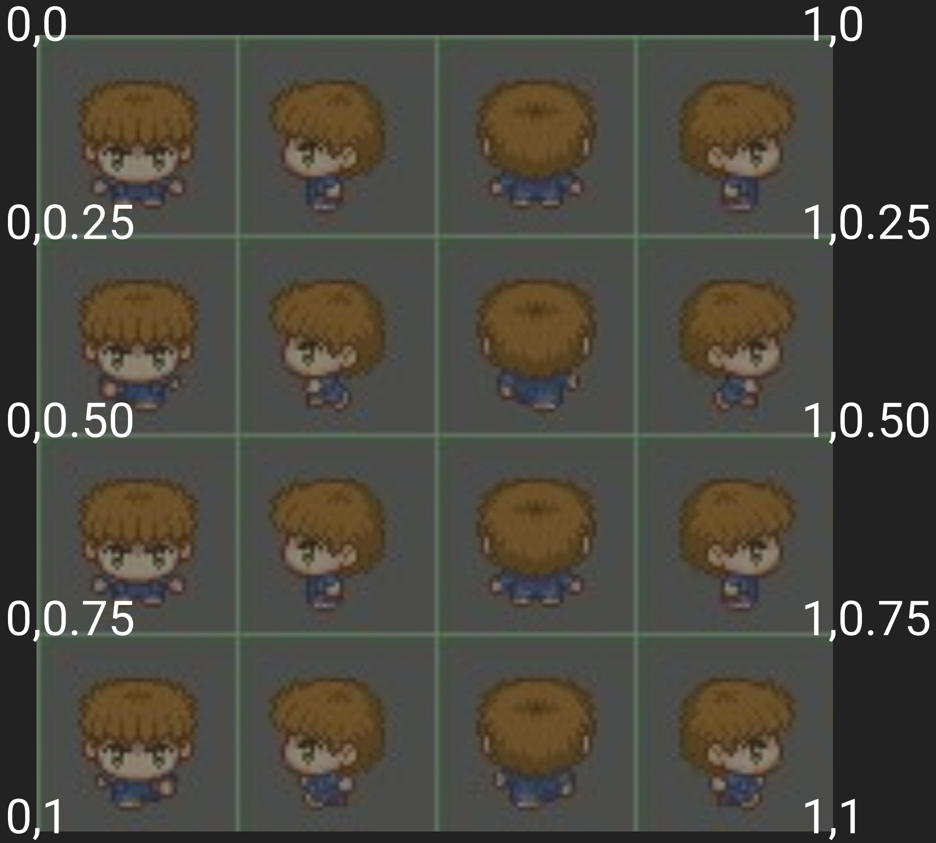
Drawing a Single Sprite

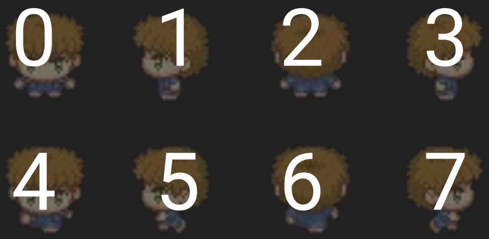
(From a Texture Atlas)

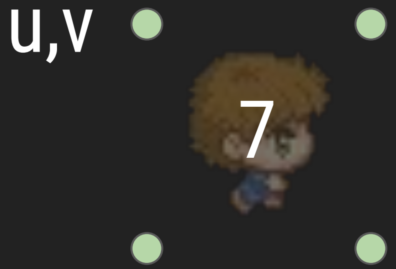
george_0.png

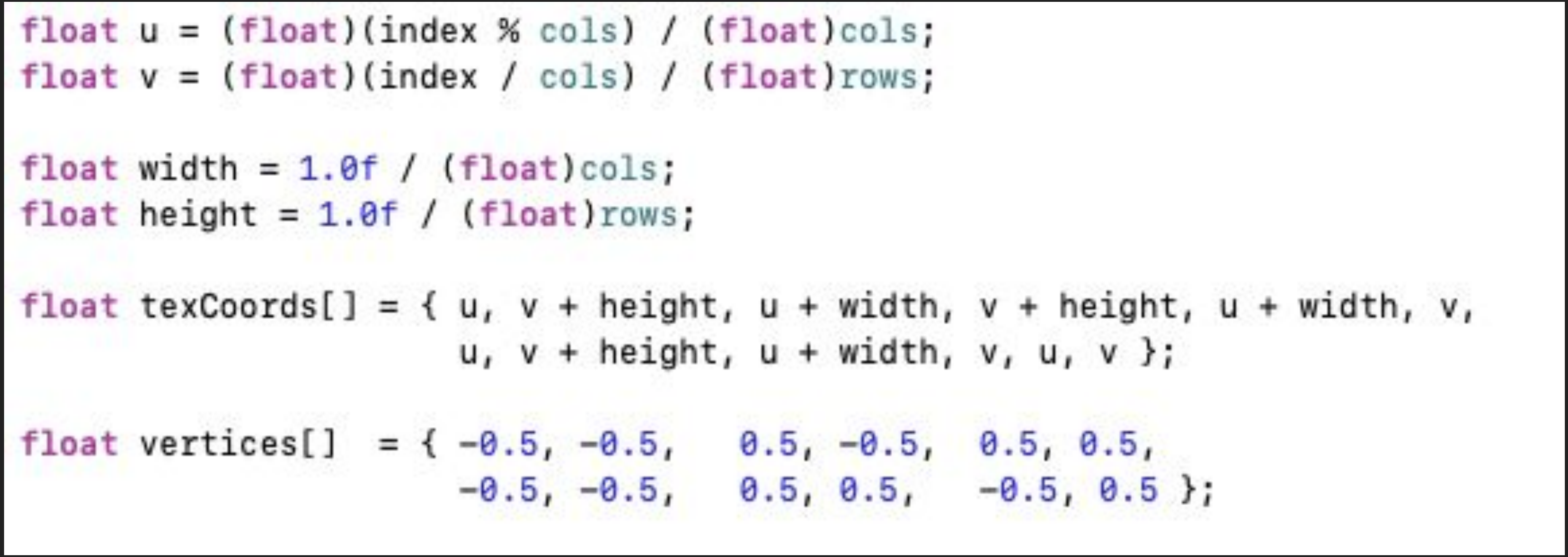


We need the UV coordinates of the individual sprite.

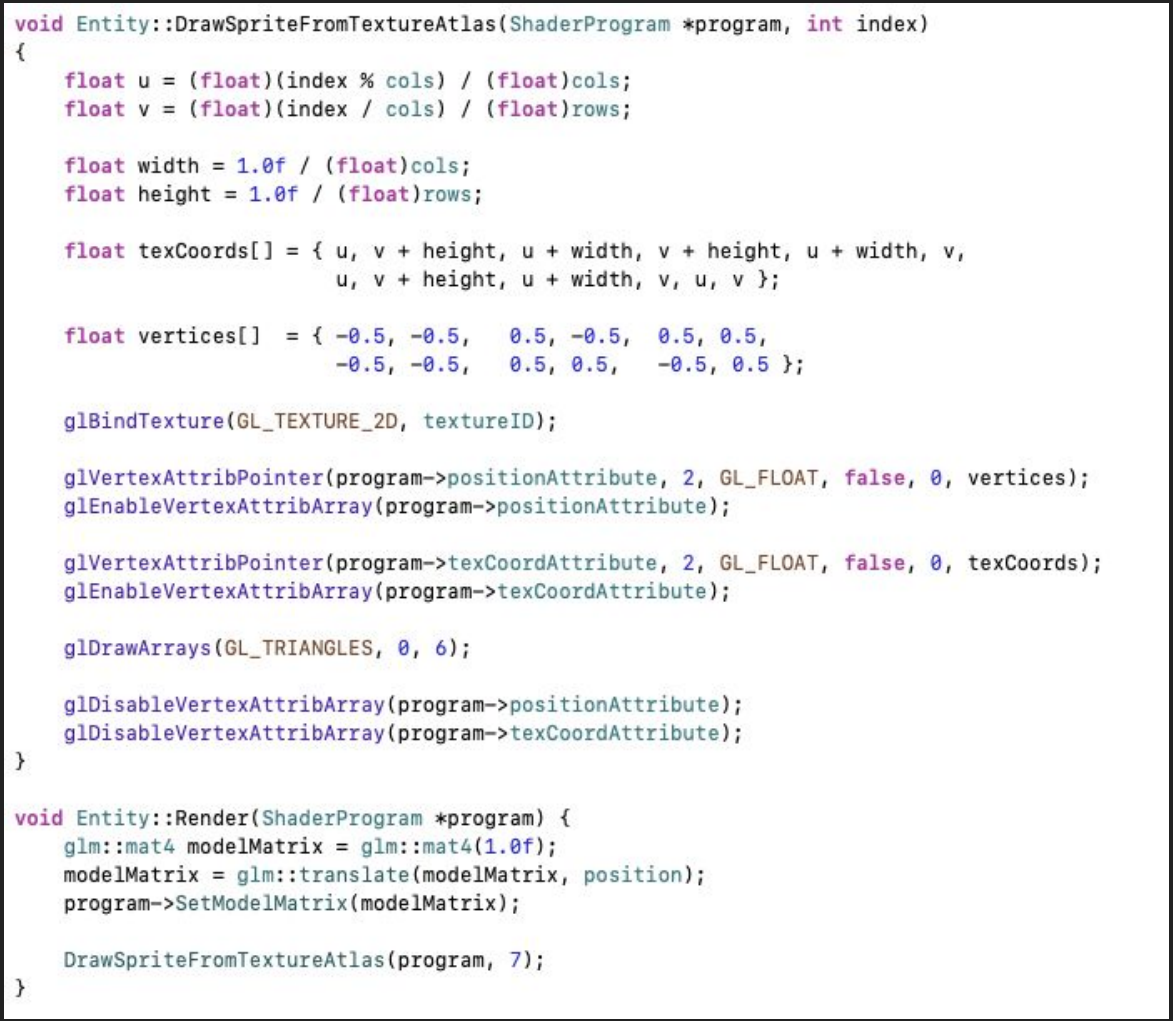






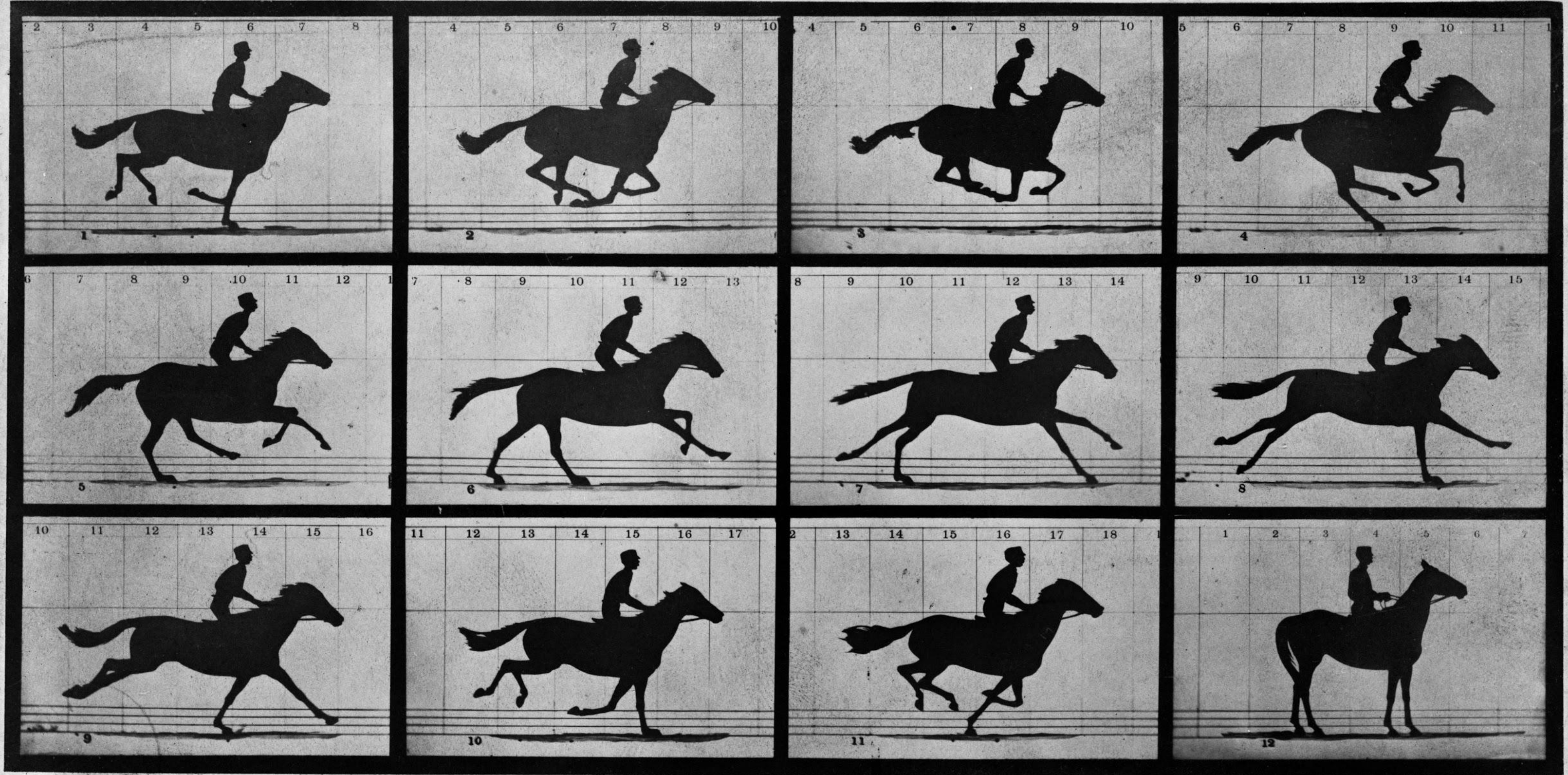


```
float u = (float)(index % cols) / (float)cols;
float v = (float)(index / cols) / (float)rows;

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 };
```



```cpp
void Entity::DrawSpriteFromTextureAtlas(ShaderProgram *program, 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 Entity::Render(ShaderProgram *program) {
    glm::mat4 modelMatrix = glm::mat4(1.0f);
    modelMatrix = glm::translate(modelMatrix, position);
    program->SetModelMatrix(modelMatrix);

    DrawSpriteFromTextureAtlas(program, 7);
}
```

Animation!



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

THE HORSE IN MOTION.

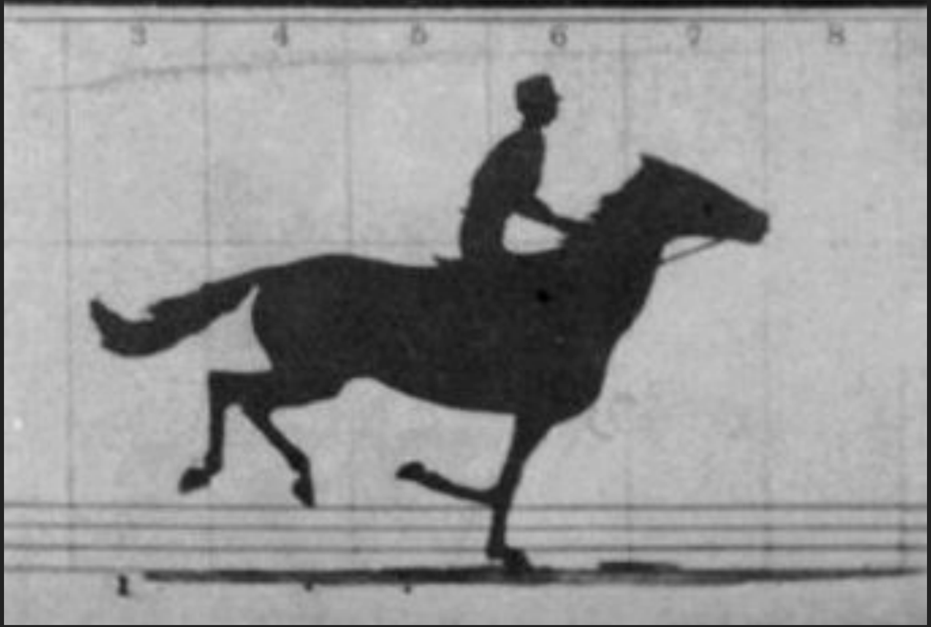
Illustrated by
MUYBRIDGE.

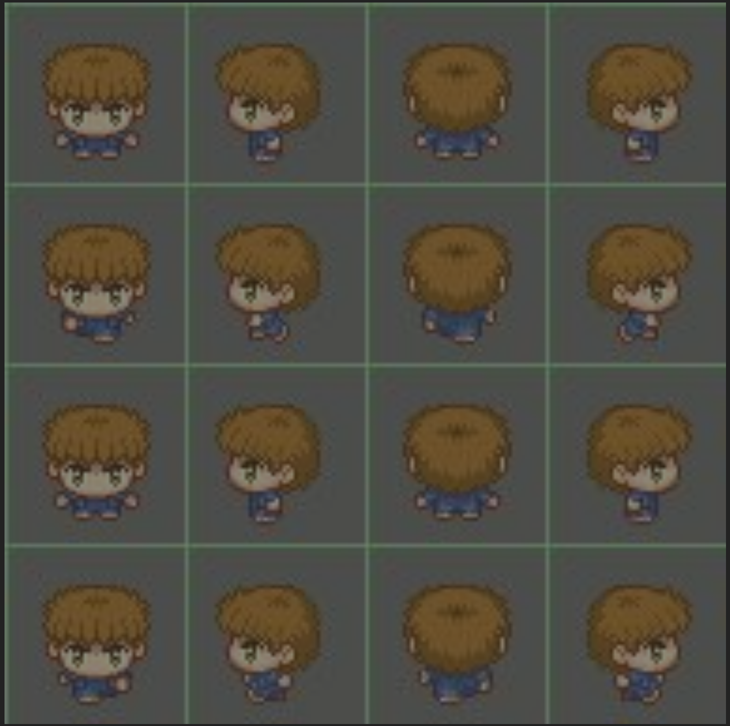
AUTOMATIC ELECTRO-PHOTOGRAPH.

[illegible]

"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
1

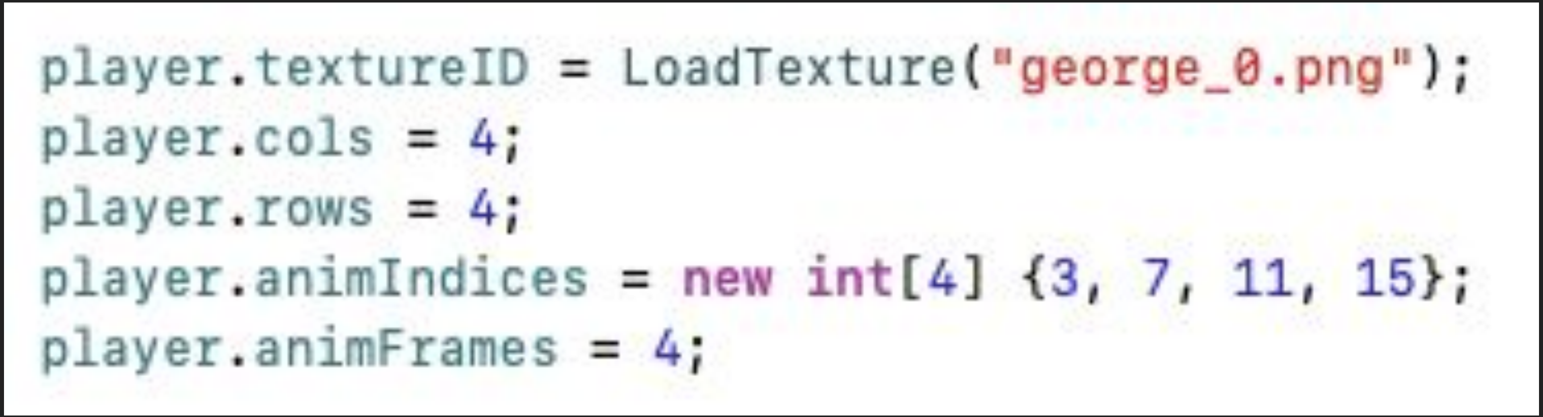


Define indices of animation: (3, 7, 11, 15)

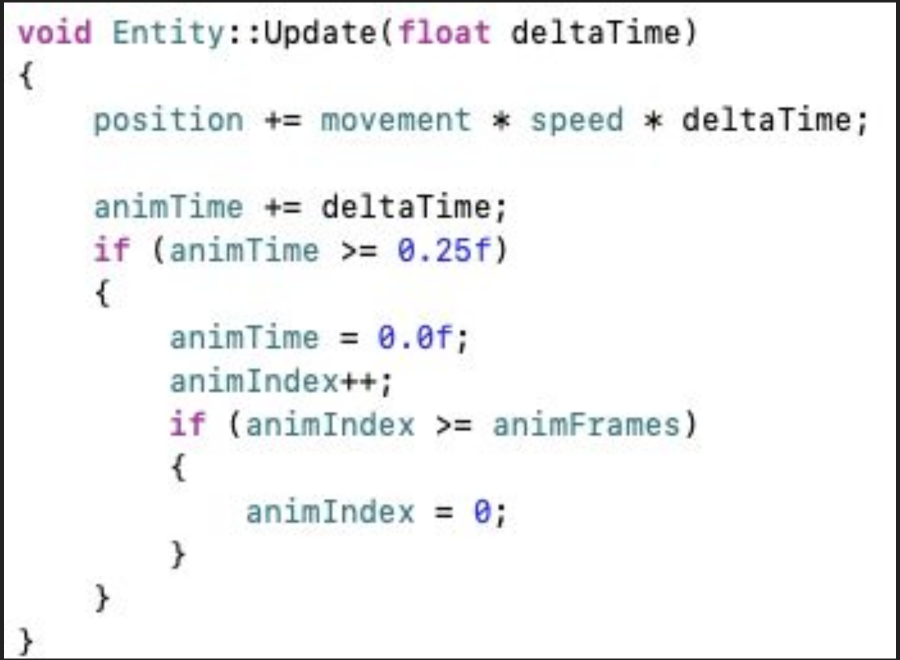
Have a timer.

Go to next frame when timer hits value.

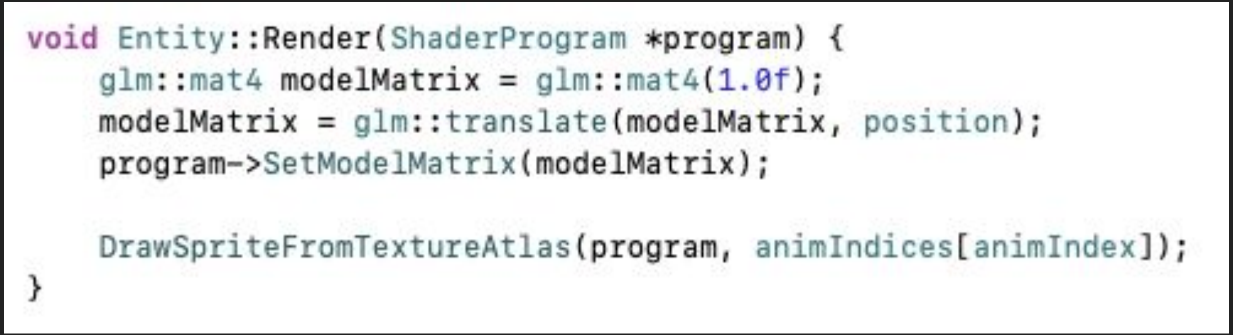
If last frame (go to first) - looping.



```
player.textureID = LoadTexture("george_0.png");
player.cols = 4;
player.rows = 4;
player.animIndices = new int[4] {3, 7, 11, 15};
player.animFrames = 4;
```



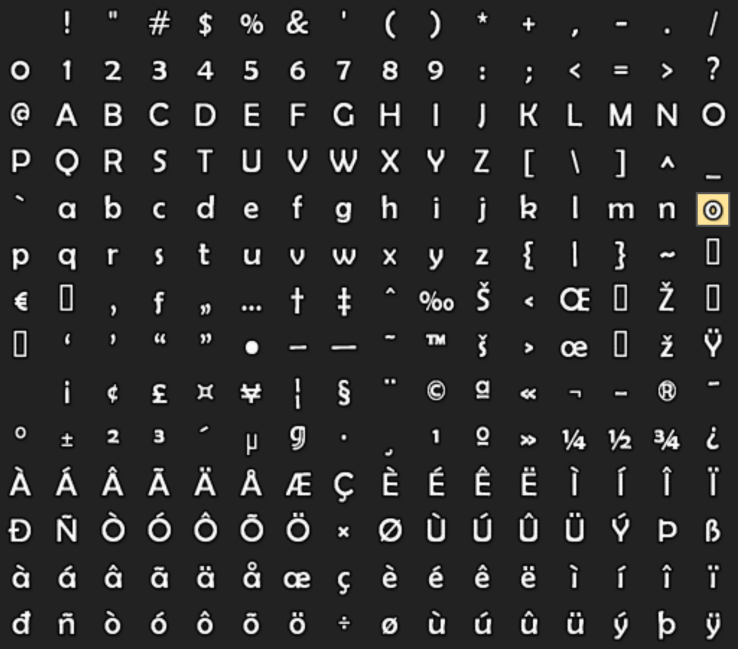
```
void Entity::Update(float deltaTime)
{
    position += movement * speed * deltaTime;

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

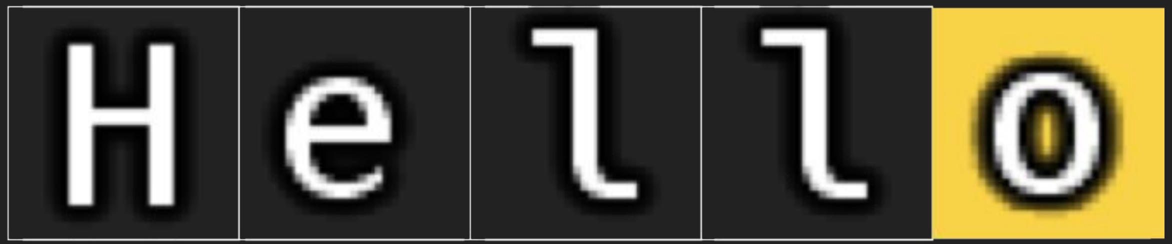


```
void Entity::Render(ShaderProgram *program) {
    glm::mat4 modelMatrix = glm::mat4(1.0f);
    modelMatrix = glm::translate(modelMatrix, position);
    program->SetModelMatrix(modelMatrix);

    DrawSpriteFromTextureAtlas(program, animIndices[animIndex]);
}
```

Monospaced Font Rendering

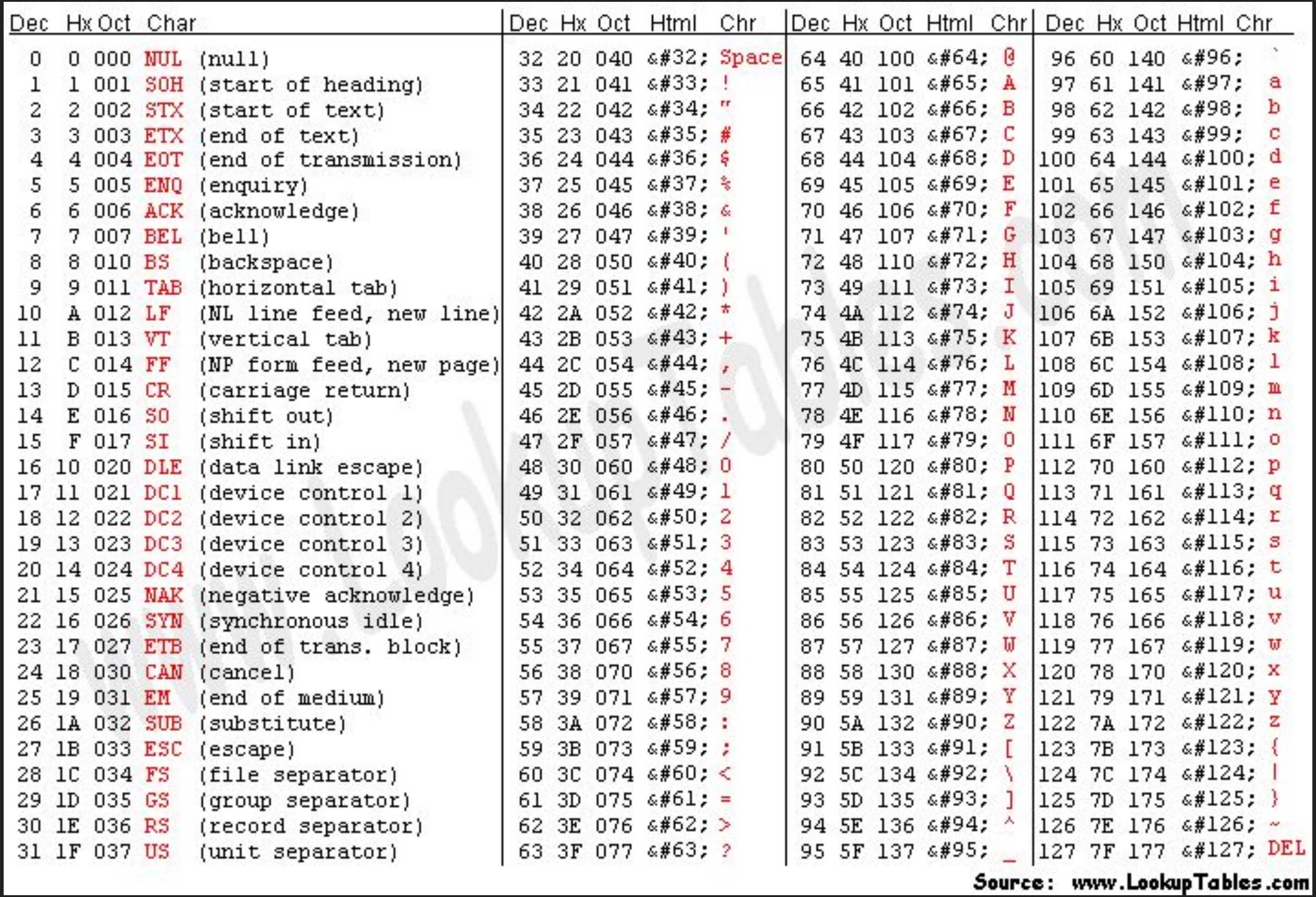
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	□
€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	Ž	□
□	‘	’	“	”	•	–	—	˜	™	š	›	œ	□	ž	Ÿ
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·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





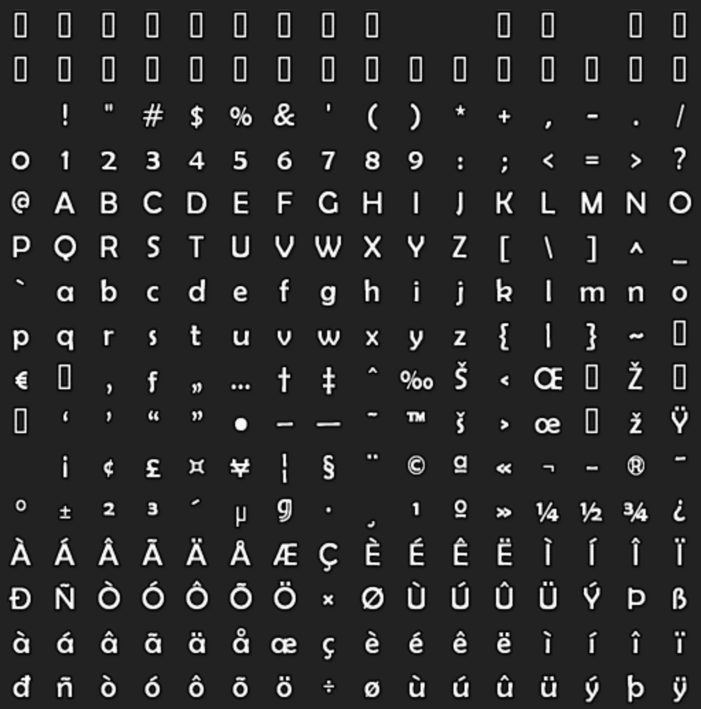
For each character in a string

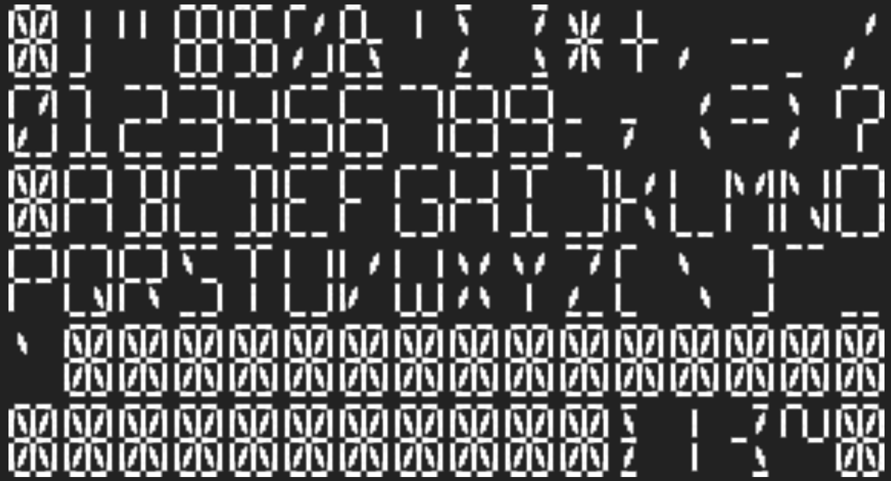
- Draw 2 Triangles
- Use UV coordinates for character



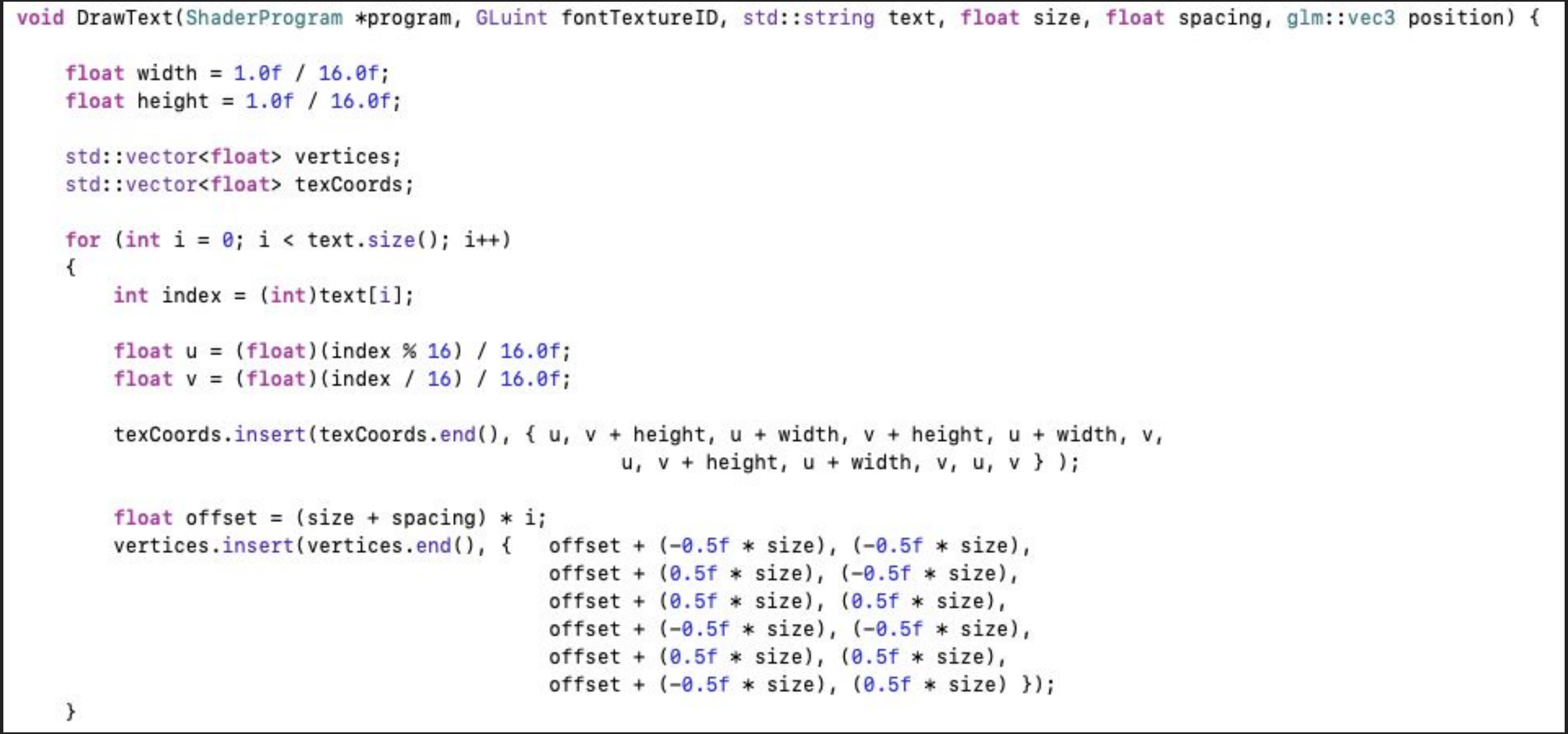
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B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

Source: www.LookupTables.com



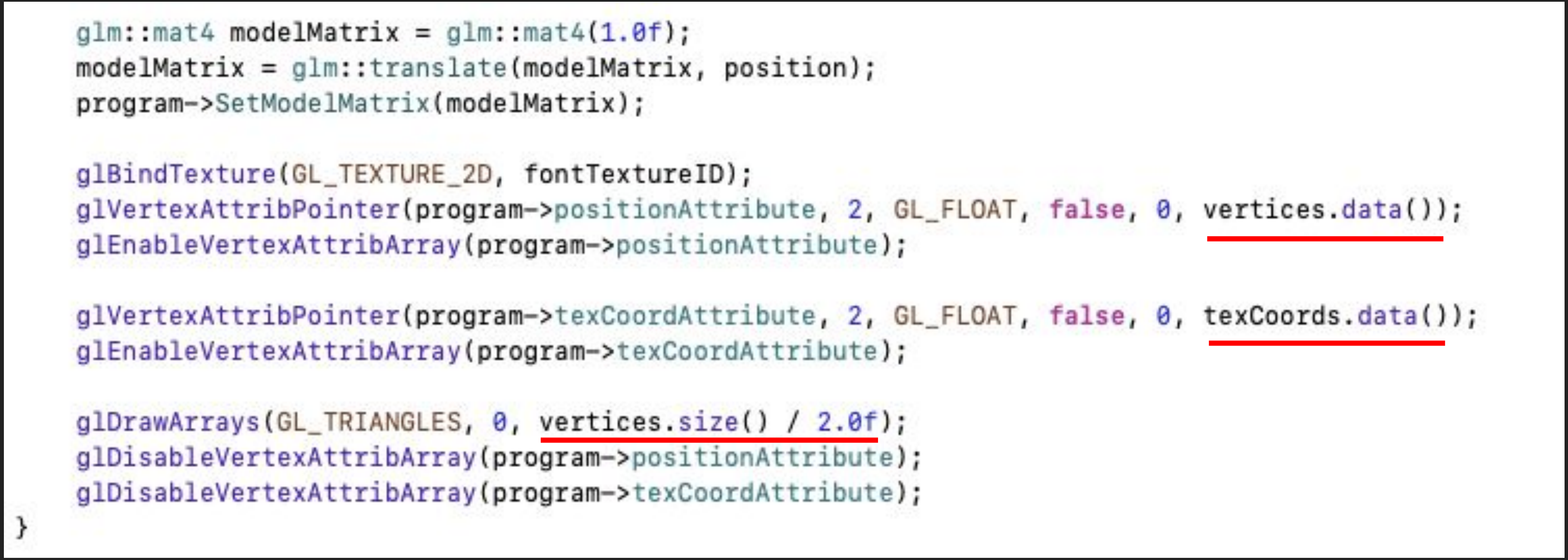


Depending on the font texture, you may have to shift the character value.



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

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

        float offset = (size + spacing) * i;
        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) });
    }
```

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);

    glBindTexture(GL_TEXTURE_2D, fontTextureID);
    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);

    glDrawArrays(GL_TRIANGLES, 0, vertices.size() / 2.0f);
    glDisableVertexAttribArray(program->positionAttribute);
    glDisableVertexAttribArray(program->texCoordAttribute);
}
```

Let’s Animate George!

(grab code from the
“Animation” example in github)