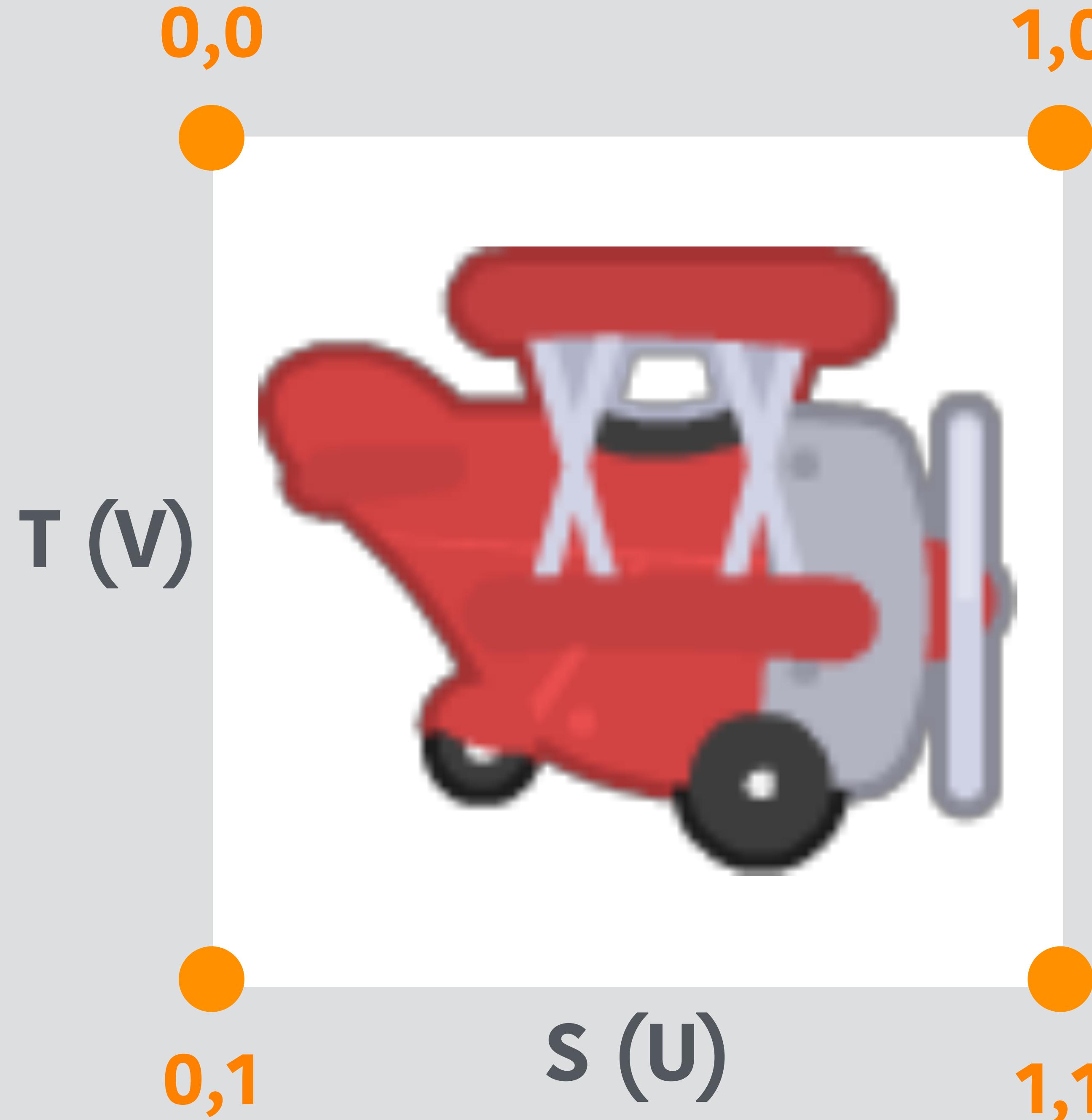


# Graphics Foundations



Part 3

# Texture coordinates

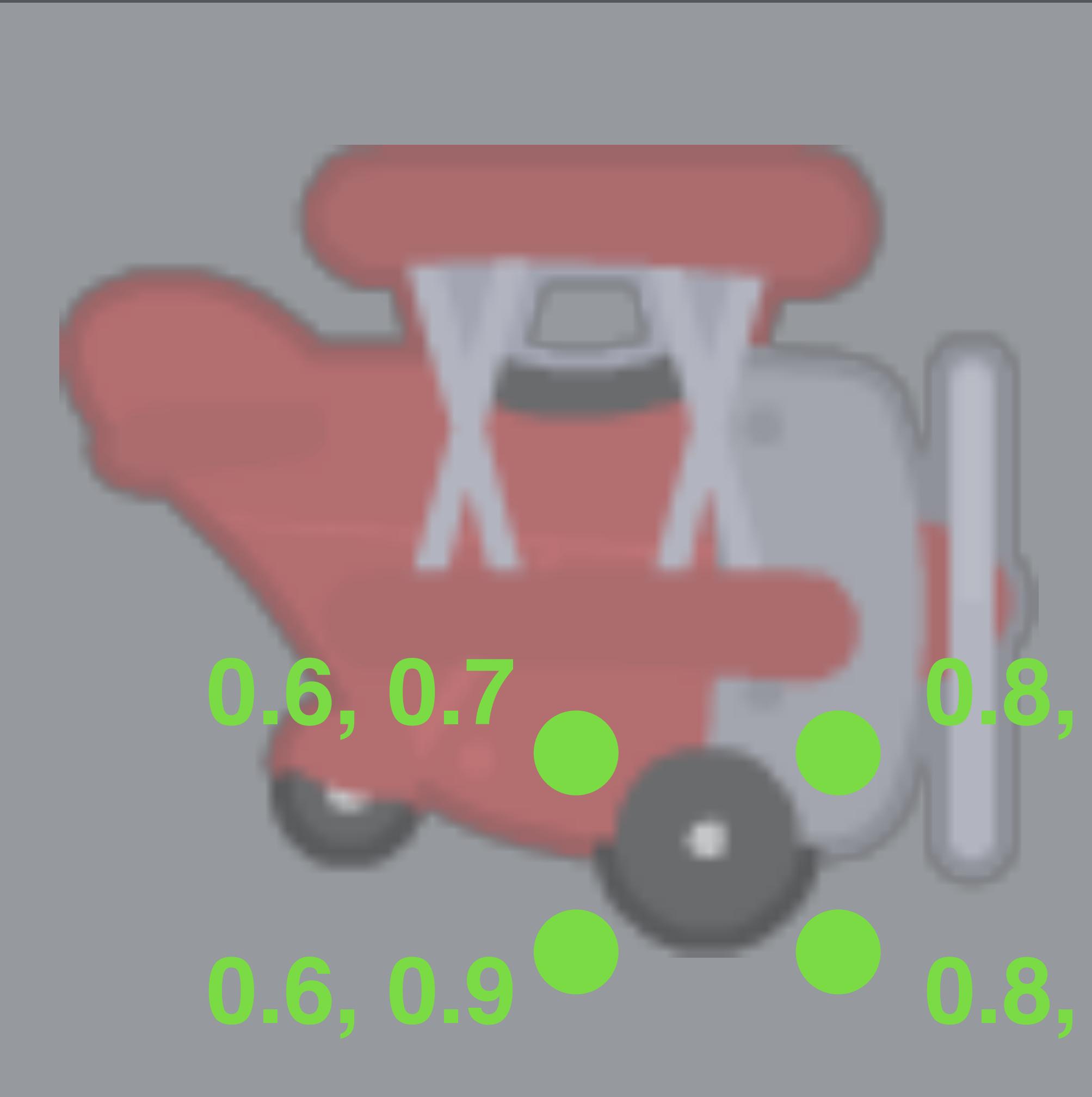


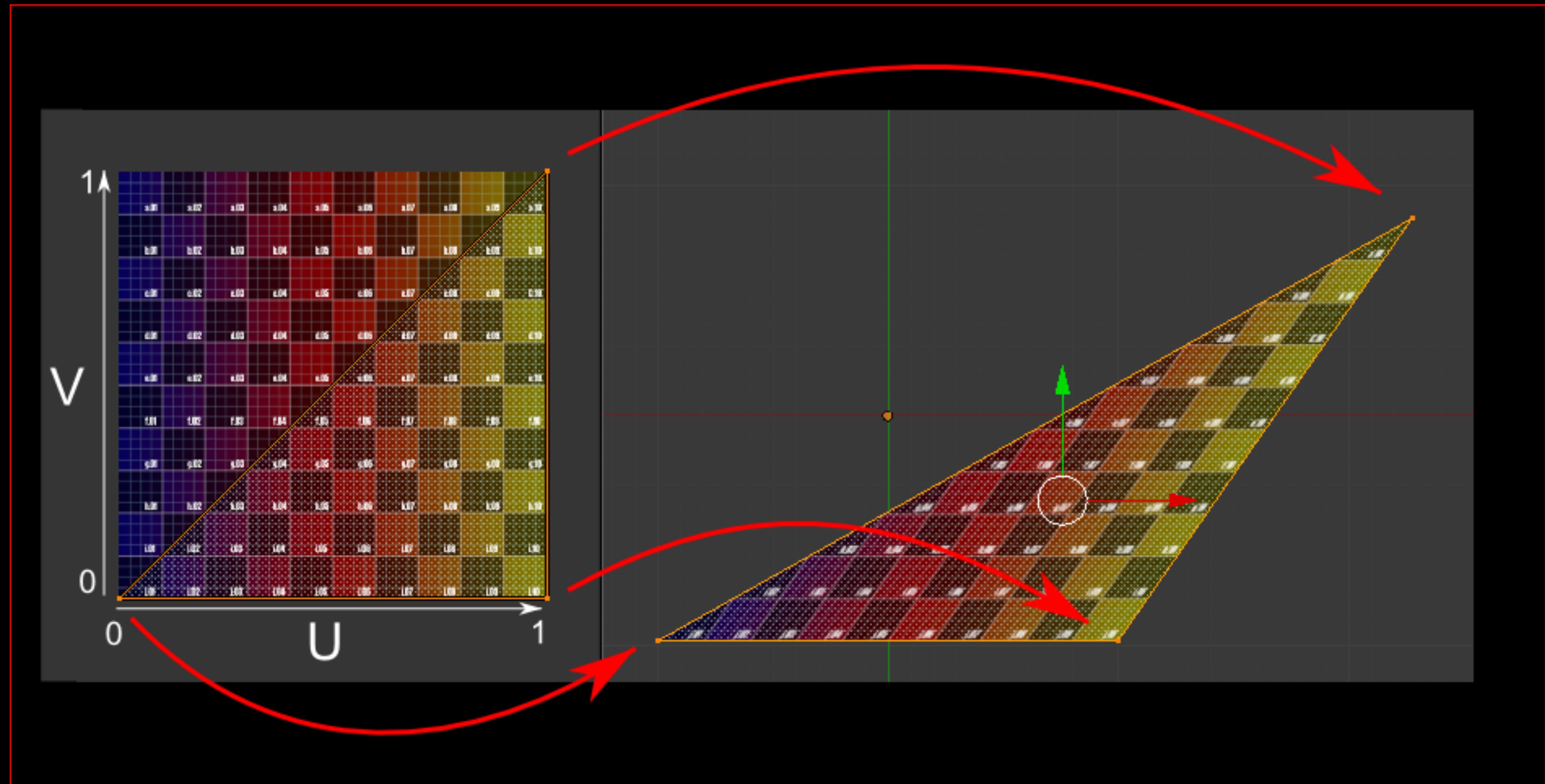
Texture coordinates  
are defined in 0-1  
units called UV  
coordinates, not  
pixels!

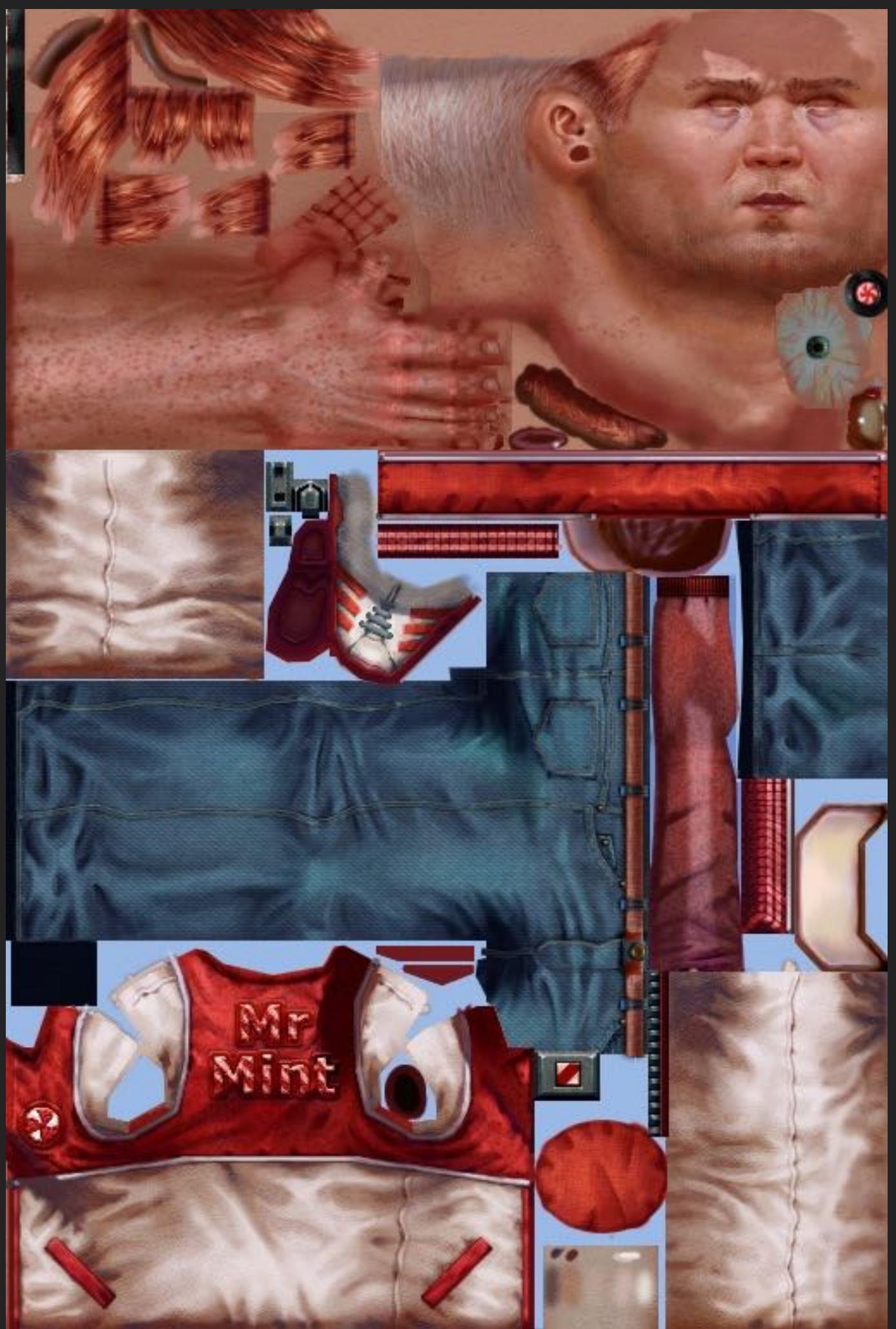
```
void glVertexAttribPointer (GLint index, GLint  
size, GLenum type, GLboolean normalized, GLsizei  
stride, const GLvoid *pointer);
```

Defines an array of **texture coordinate data**.

```
float texCoords[] = {0.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f,  
1.0f, 0.0f, 0.0f, 0.0f};  
glVertexAttribPointer(program.texCoordAttribute, 2, GL_FLOAT, false, 0, texCoords);
```

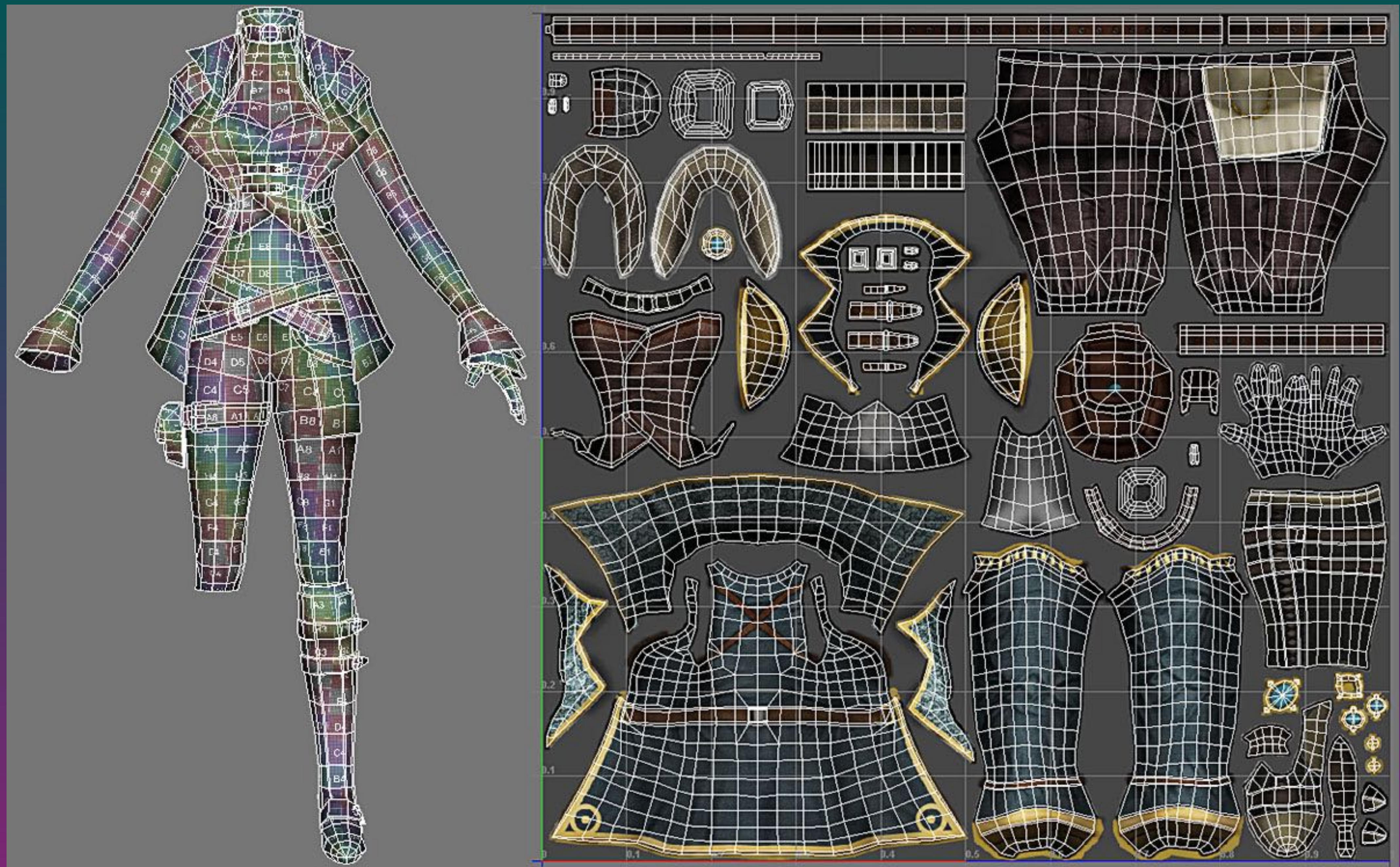






Ben Mathis  
[www.poopinmymouth.com](http://www.poopinmymouth.com)



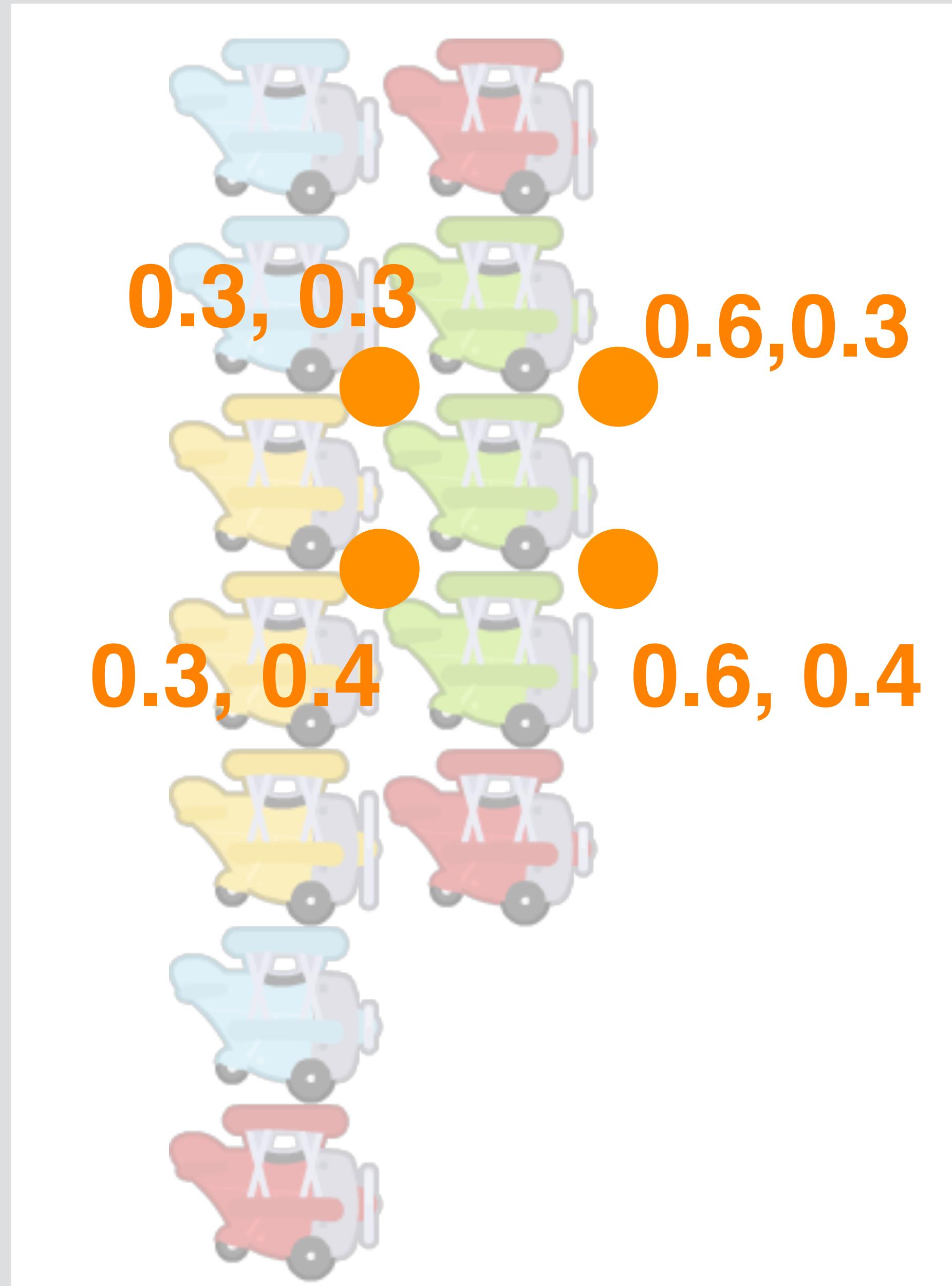


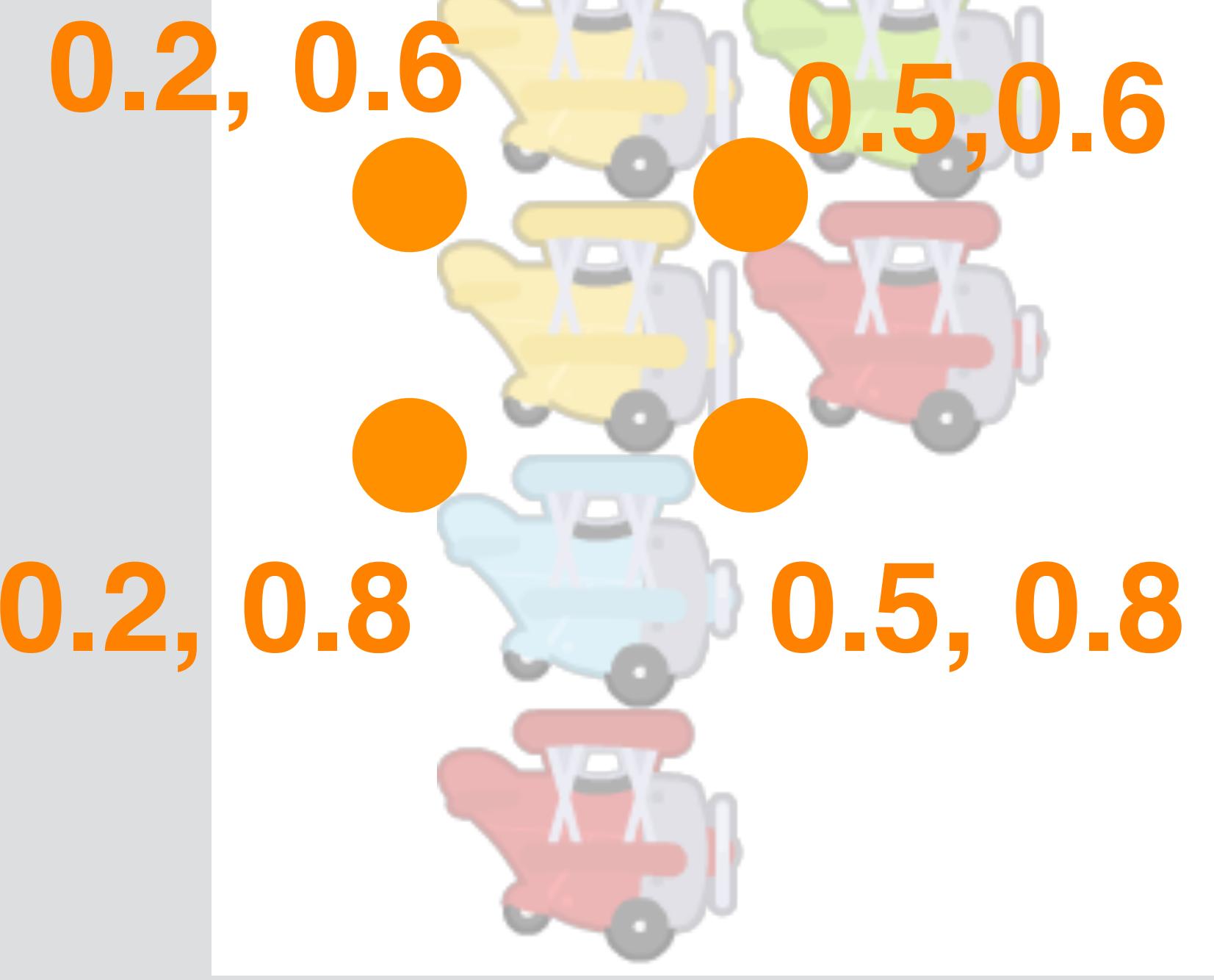
# Using texture coordinates in 2D graphics.

# Texture atlases.



A single texture that contains multiple sprites arranged in a single image.











□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □  
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □  
! " # \$ % & ' ( ) \* + , - . /  
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 " .. † ‡ ^ ‰ Š < € □ Ž □  
□ , " " • – – „ ™ Š > œ □ ž Ÿ  
i ¢ £ ₧ ¥ ¤ § ¨ © ª « ¬ - ® ¯  
° ± ² ³ ´ μ ¶ ¤ · ¨ ¹ º » ¼ ½ ¾ ð  
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

# Evenly spaced sprite sheets









```
int index = 10;
int spriteCountX = 8;
int spriteCountY = 4;
float u = (float)((int)index) % spriteCountX / (float) spriteCountX;
float v = (float)((int)index) / spriteCountX / (float) spriteCountY;
float spriteWidth = 1.0/(float)spriteCountX;
float spriteHeight = 1.0/(float)spriteCountY;

GLfloat spriteUVs[] = { u, v,
                        u, v+spriteHeight,
                        u+spriteWidth, v+spriteHeight,
                        u+spriteWidth, v
};
```

```
void DrawSpriteSheetSprite(ShaderProgram &program, int index, int spriteCountX,
int spriteCountY) {

    float u = (float)((int)index) % spriteCountX / (float) spriteCountX;
    float v = (float)((int)index) / spriteCountX / (float) spriteCountY;
    float spriteWidth = 1.0/(float)spriteCountX;
    float spriteHeight = 1.0/(float)spriteCountY;

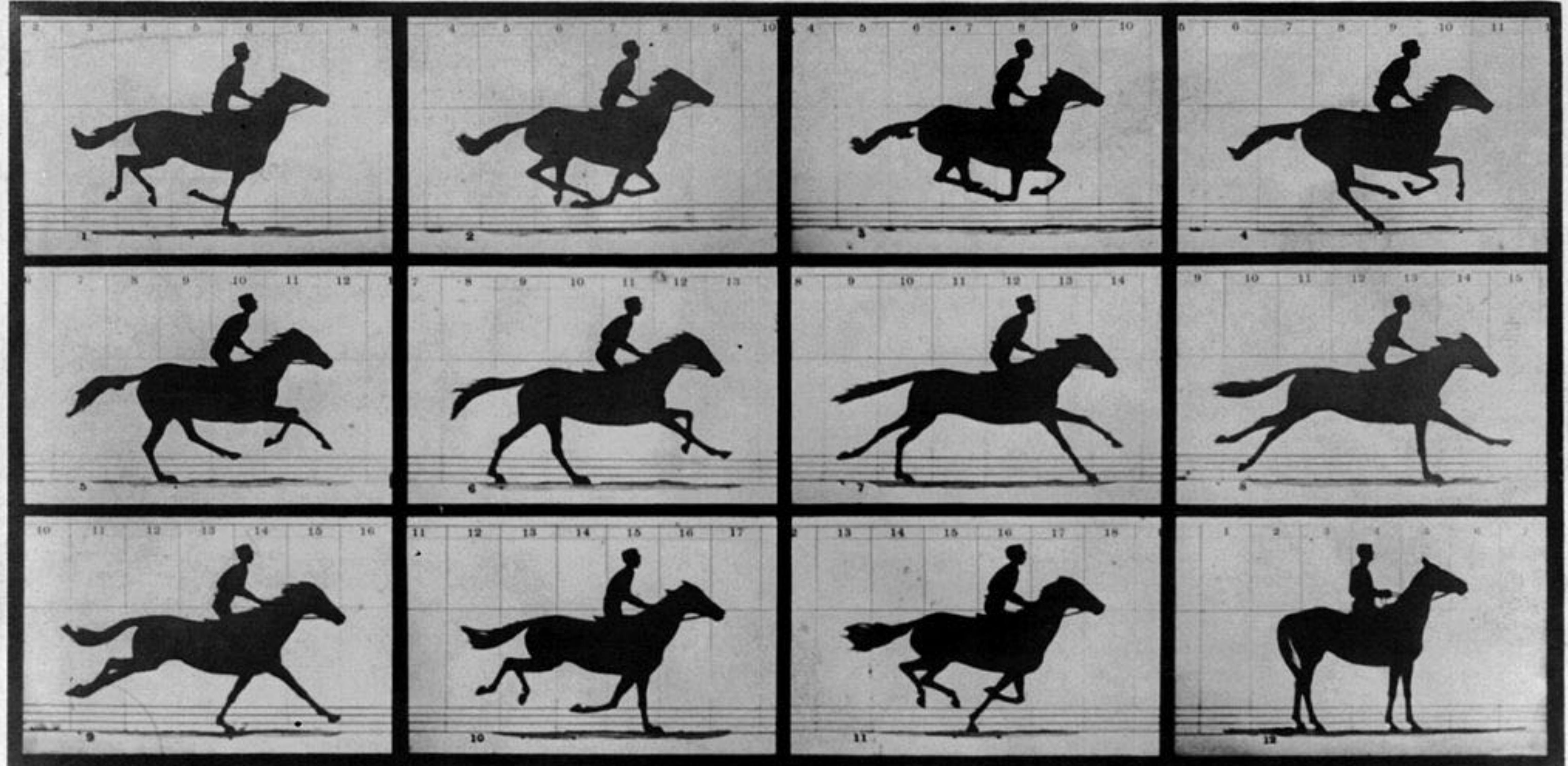
    float texCoords[] = {
        u, v+spriteHeight,
        u+spriteWidth, v,
        u, v,
        u+spriteWidth, v,
        u, v+spriteHeight,
        u+spriteWidth, v+spriteHeight
    };

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

    // draw this data

}
```

# Sprite animation



Copyright, 1878, by MUYBRIDGE.

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

## THE HORSE IN MOTION.

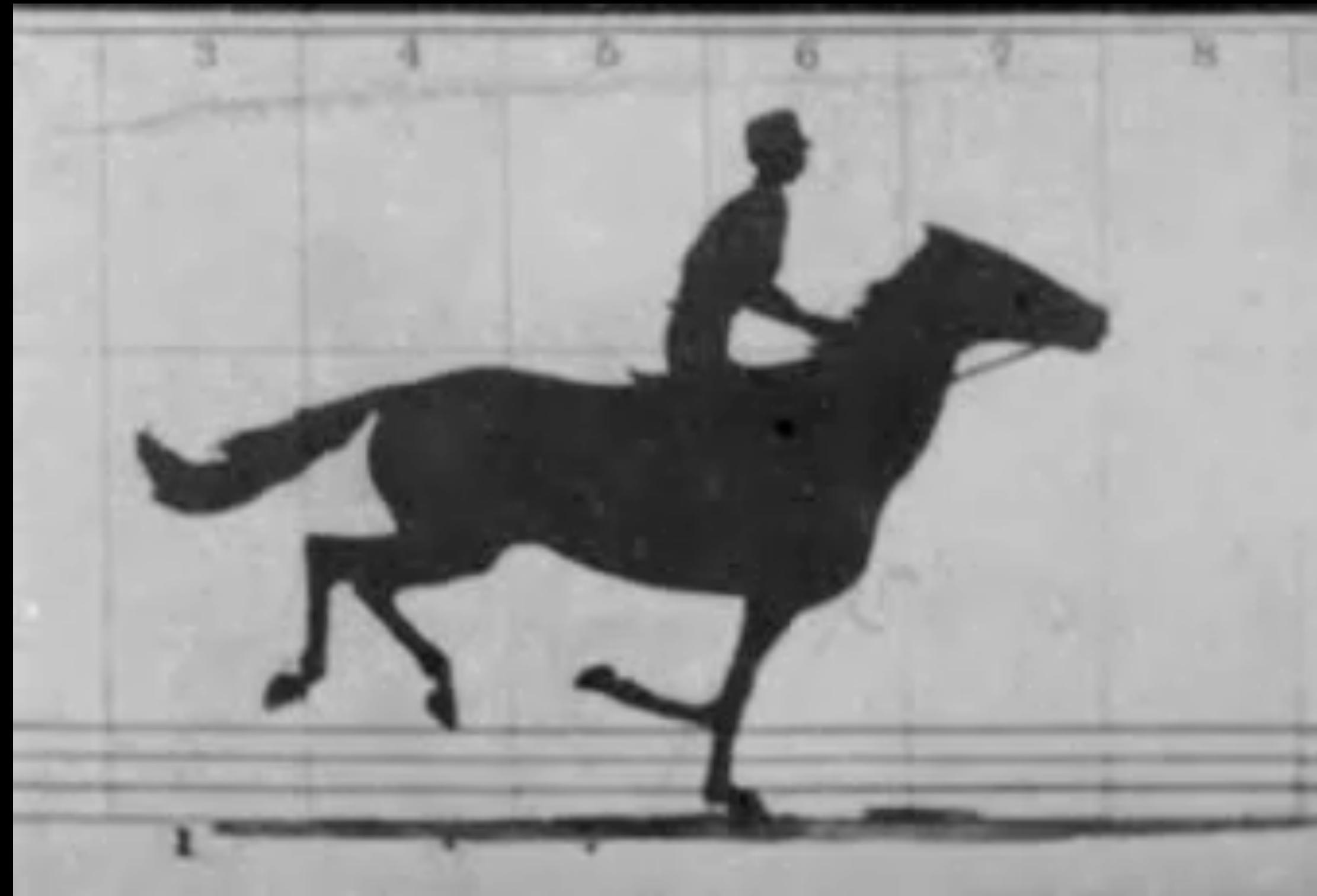
Illustrated by

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.





1. Define indices of an animation (e.g. 0-6)
2. Keep a timer
3. Go to next frame when timer hits desired value.
4. If at the last frame, go to first frame (if looped animation).

```
const int runAnimation[] = {9, 10, 11, 12, 13};  
const int numFrames = 5;  
float animationElapsed = 0.0f;  
float framesPerSecond = 30.0f;  
int currentIndex = 0;
```

In our **loop**:

```
animationElapsed += elapsed;  
  
if(animationElapsed > 1.0/framesPerSecond) {  
    currentIndex++;  
    animationElapsed = 0.0;  
  
    if(currentIndex > numFrames-1) {  
        currentIndex = 0;  
    }  
}  
  
DrawSpriteSheetSprite(runAnimation[currentIndex], 8, 4);
```

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 { | } ~ □  
€ □ ; f " .. † ‡ ^ ‰ Š < € □ Ž □  
□ , " " • – – „ ™ Š > œ □ ž Ÿ  
i ¢ £ ₧ ¥ ¤ § ¨ © ª « ¬ - ® ¯  
° ± ² ³ ´ μ ¶ ¤ · ¨ ¹ º » ¼ ½ ¾ ð  
À Á Â Ã Ä Å Æ Ç Ě 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

|   |   |   |    |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|----|---|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| □ | □ | □ | □  | □ | □   | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |   |
| ! | " | # | \$ | % | &   | ' | ( | ) | * | + | , | - | . | / |   |   |   |   |   |   |
| 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 | ó |   |   |   |   |   |
| p | q | r | s  | t | u   | v | w | x | y | z | { | } |   |   | ~ | □ |   |   |   |   |
| € | ; | f | "  | " | ... | † | ‡ | ^ | ‰ | Š | < | € | □ | Ž | □ |   |   |   |   |   |
| ‘ | ’ | “ | ”  | • | —   | — | ~ | ™ | š | > | æ | □ | ž | ÿ |   |   |   |   |   |   |
| ı | ¢ | £ | ¤  | ¥ | ₩   | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ | ₪ |
| ◦ | ± | ² | ³  | ² | ³   | ² | ³ | ² | ³ | ² | ³ | ² | ³ | ² | ³ | ² | ³ | ² | ³ | ² |
| À | Á | Ã | Ä  | Å | Æ   | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï | Ӯ |   |   |   |   |   |
| Đ | Ñ | Ò | Ó  | Ô | Õ   | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |   |   |   |   |   |
| à | á | 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 |   |   |   |   |   |

To render a string, we must look at it character by character and draw 2 triangles for each letter using the appropriate UV coordinates.

H e l l o

```

string test = "This is a string!";
cout << (int)test[3] << endl; // prints 115, which is 's'

```

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

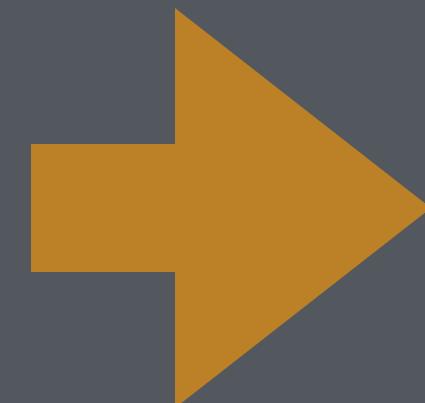
```
void DrawText(ShaderProgram &program, int fontTexture, std::string text, float size, float spacing) {  
    float character_size = 1.0/16.0f;  
  
    std::vector<float> vertexData;  
    std::vector<float> texCoordData;  
  
    for(int i=0; i < text.size(); i++) {  
        int spriteIndex = (int)text[i];  
  
        float texture_x = (float)(spriteIndex % 16) / 16.0f;  
        float texture_y = (float)(spriteIndex / 16) / 16.0f;  
  
        vertexData.insert(vertexData.end(), {  
            ((size+spacing) * i) + (-0.5f * size), 0.5f * size,  
            ((size+spacing) * i) + (-0.5f * size), -0.5f * size,  
            ((size+spacing) * i) + (0.5f * size), 0.5f * size,  
            ((size+spacing) * i) + (0.5f * size), -0.5f * size,  
            ((size+spacing) * i) + (0.5f * size), 0.5f * size,  
            ((size+spacing) * i) + (-0.5f * size), -0.5f * size,  
        });  
        texCoordData.insert(texCoordData.end(), {  
            texture_x, texture_y,  
            texture_x, texture_y + character_size,  
            texture_x + character_size, texture_y,  
            texture_x + character_size, texture_y + character_size,  
            texture_x + character_size, texture_y,  
            texture_x, texture_y + character_size,  
        });  
    }  
  
    glBindTexture(GL_TEXTURE_2D, fontTexture);  
  
    // draw this data (use the .data() method of std::vector to get pointer to data)  
    // draw this yourself, use text.size() * 6 or vertexData.size()/2 to get number of vertices  
}
```

# Texture wrap modes.

0,0



1,0



1,1

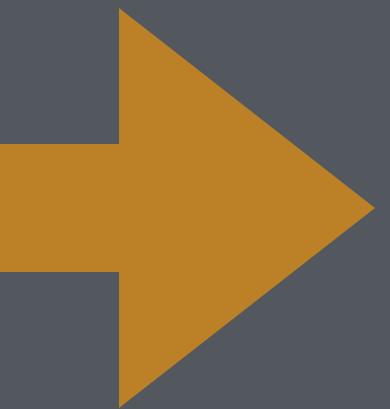


0,1



2,0

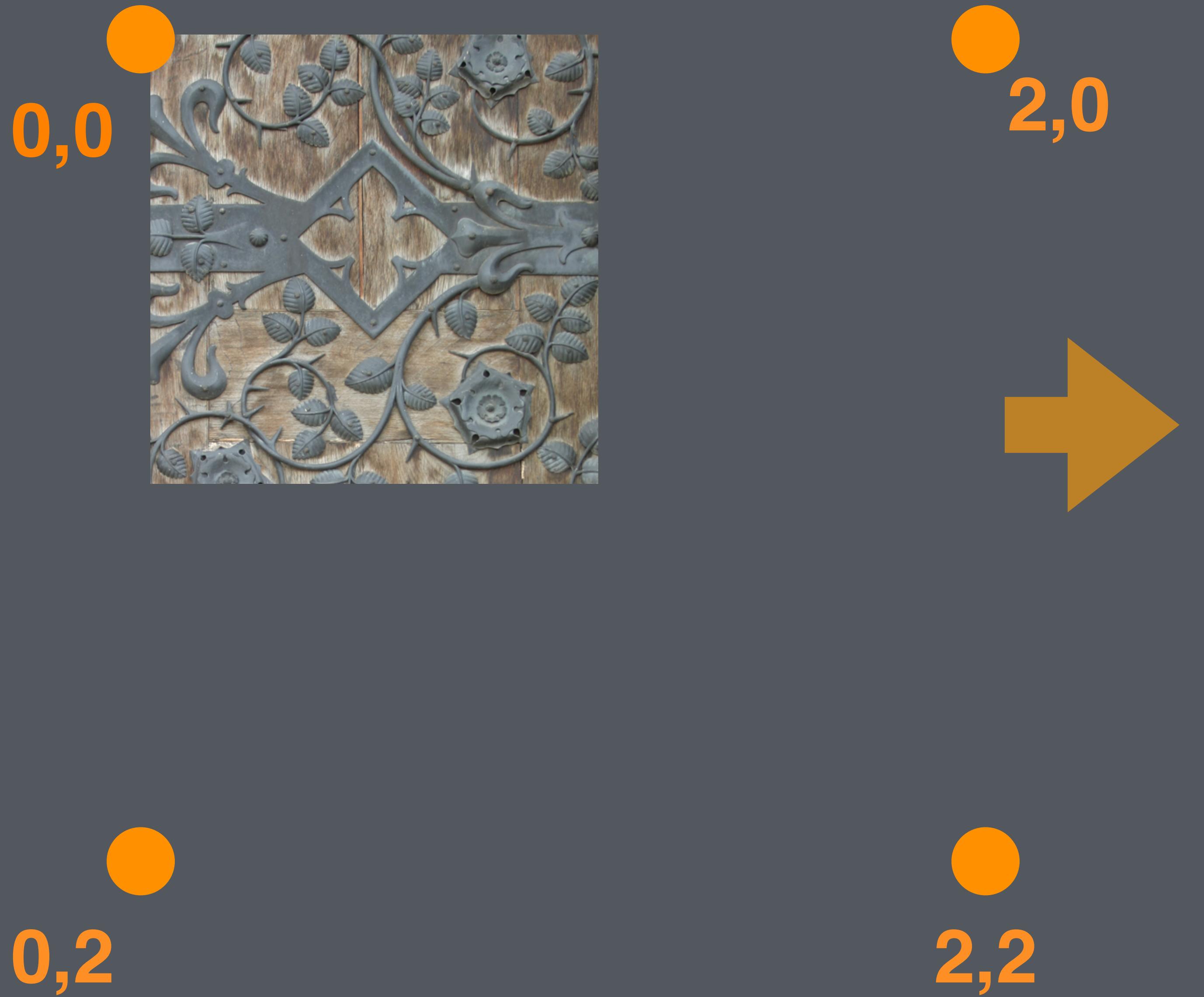
0,0



0,2

2,2

# Repeat



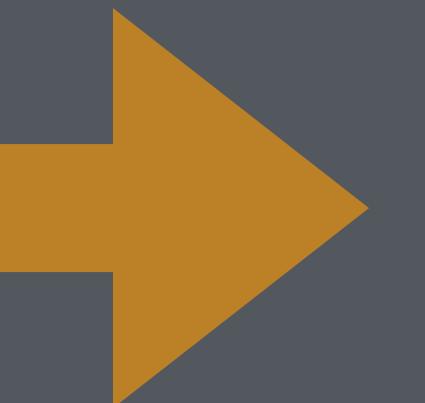
# Clamp



0,2

2,2

2,0



0,0

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

Sets a texture parameter of the specified texture target.

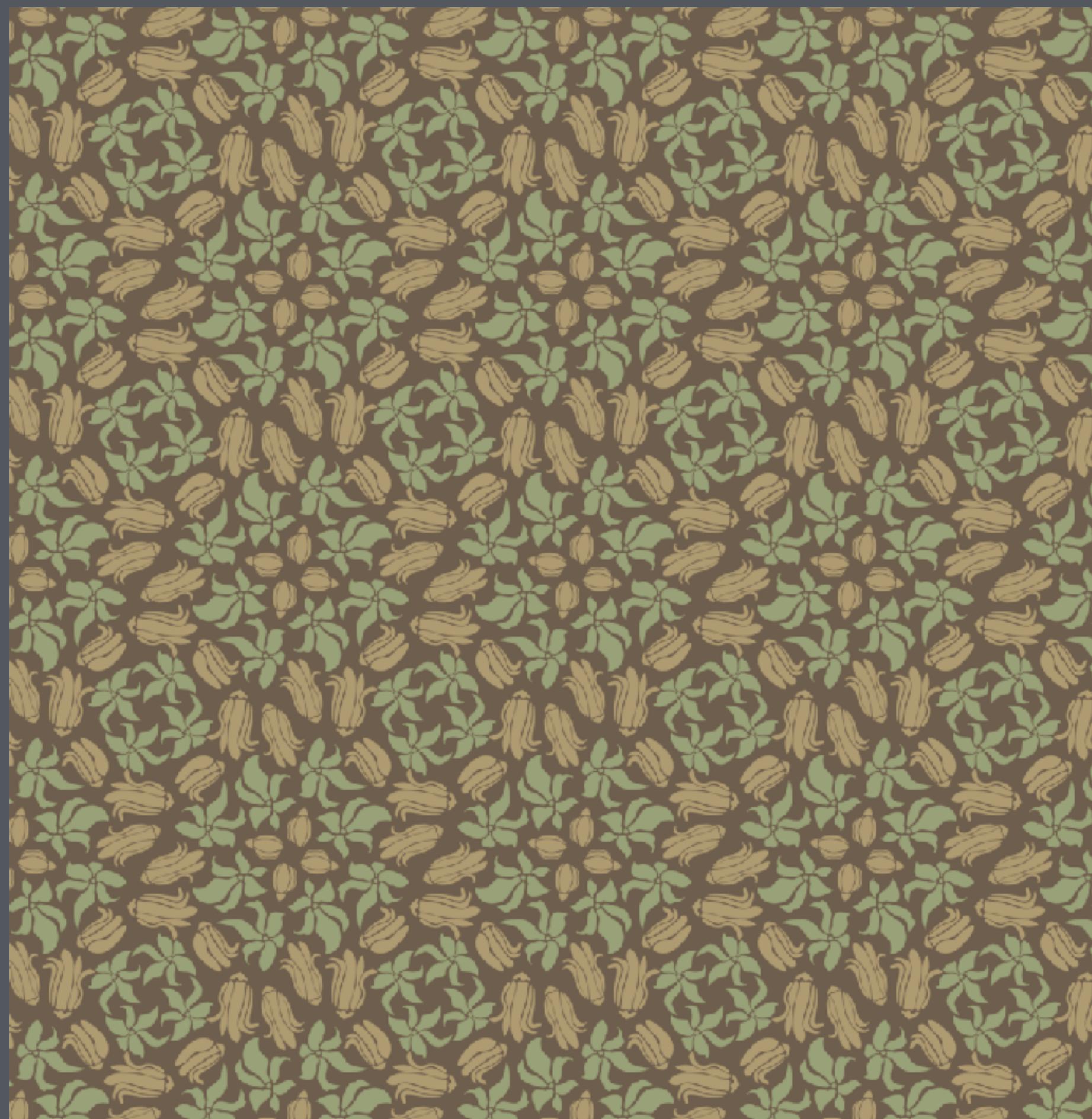
## CLAMPING

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
```

## REPEATING

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

# Use GL\_REPEAT for tiling textures



Use GL\_CLAMP for non-tiled images with alpha.