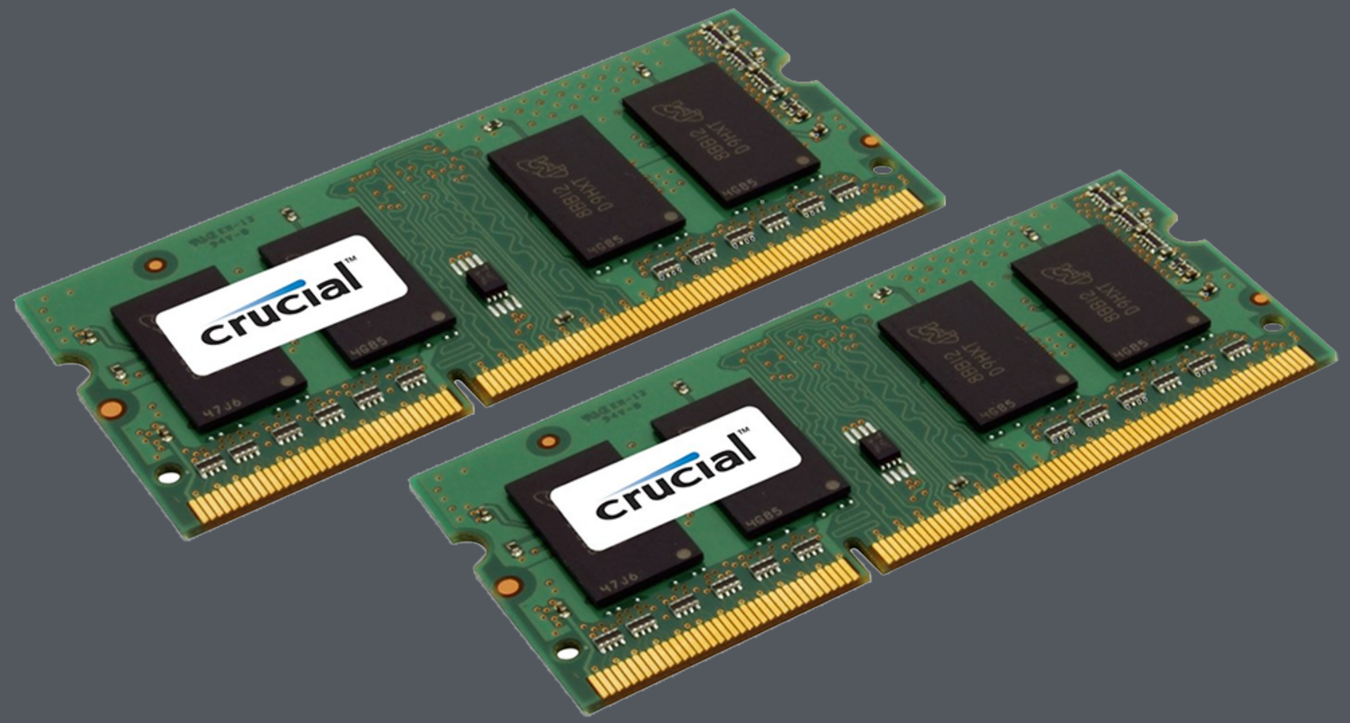
# Graphics Foundations

# Quick **pointer** refresher




| 1 | 0 | 0 | 1 |
|---|---|---|---|
| 0 | 1 | 0 | 1 |

0 1 2 3 4 5 6 7 8 9 ... 7999999996 7999999997 7999999998 7999999999

```
char myVar = ‘a’; // char is an 8-bit variable
```


```
float myVar = 1.0f; // float is a 32-bit variable
```


```
double myVar = 1.0; // double is a 64-bit variable
```

```
float myVar = 16.0f;
```


```
float *myVarPtr = &myVar;

cout << myVarPtr << endl; // prints 3
```

```
float myVar = 16.0f;

float *myVarPtr = &myVar;

cout << myVarPtr << endl; // prints 3

cout << *myVarPtr << endl; // prints 16

cout << &myVar << endl; // prints 3
```

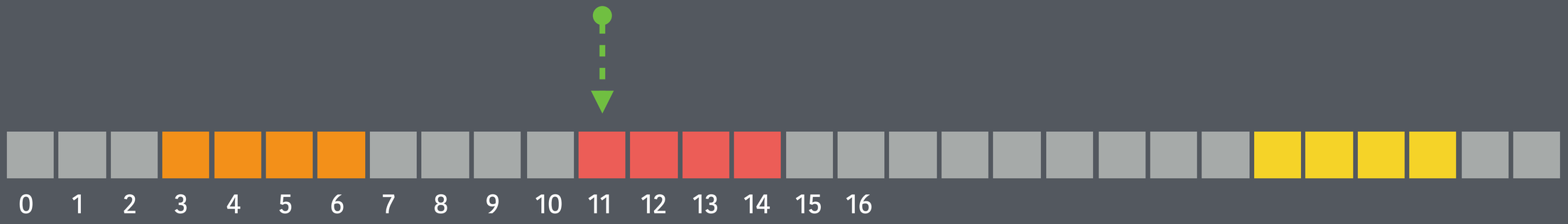
Reference

Dereference

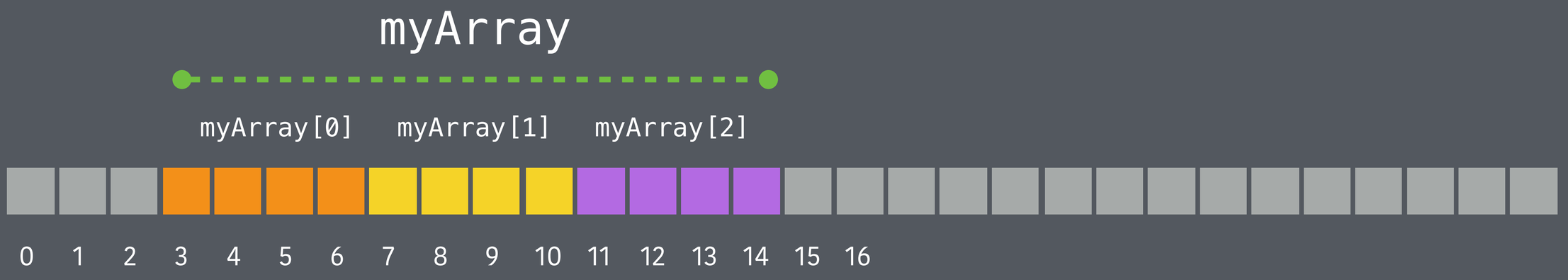
```
float myVar = 16.0f;

float *myVarPtr = &myVar;

cout << myVarPtr; // prints 3

cout << *myVarPtr; // prints 16

cout << &myVar; // prints 3

float **myVarPtrPtr = &myVarPtr;

cout << myVarPtrPtr; // prints 11
```

Reference

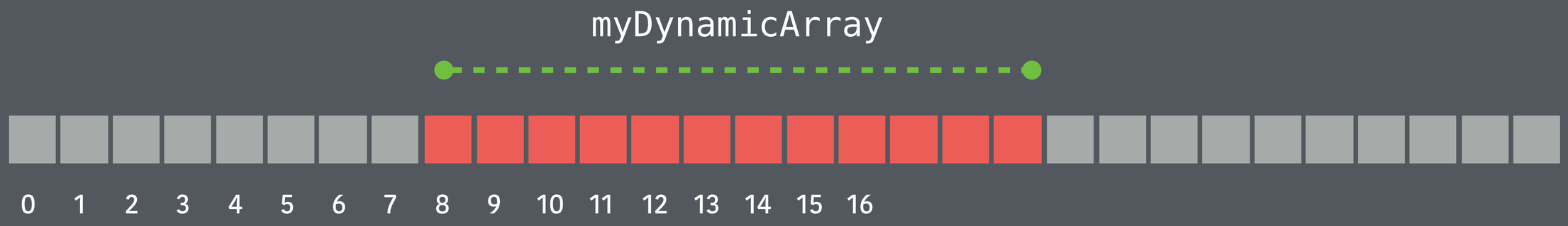
Dereference



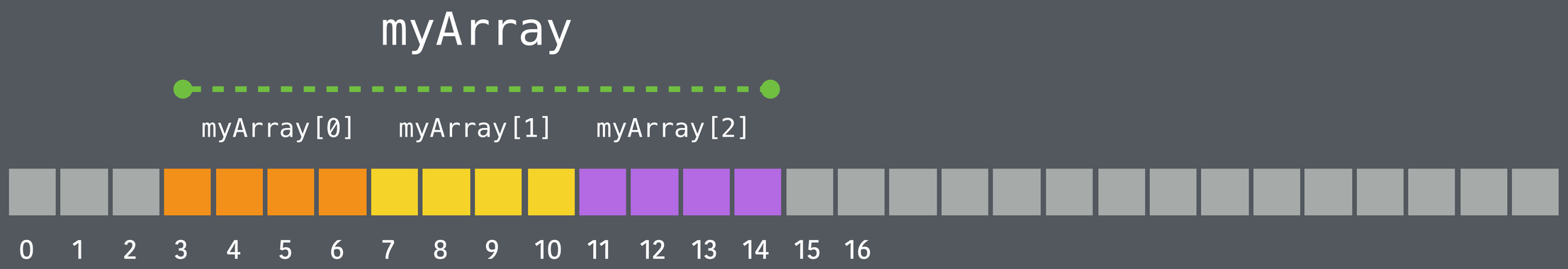
```
float myArray[3] = {13.5f, 2.3f, 5.4f};

cout << myArray; // prints 3
cout << &myArray[0]; // prints 3

float *myArrayAsAPointer = myArray;
cout << myArrayAsAPointer; // prints 3

cout << myArrayAsAPointer[0]; // prints 13.5
cout << myArrayAsAPointer[1]; // prints 2.3
```



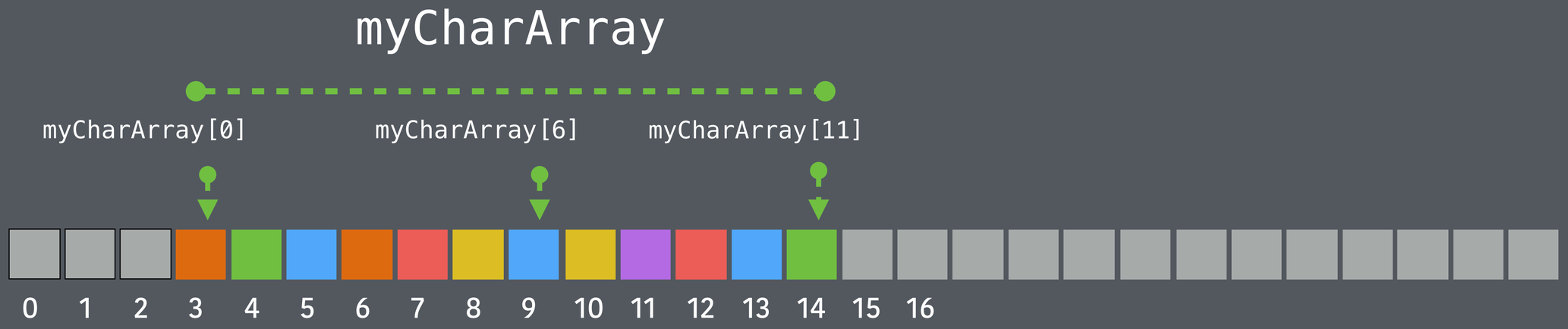
```
float *myDynamicArray;

cout << myDynamicArray; // 0x0 or some uninitialized location
cout << myDynamicArray[0]; // crash or garbage!

myDynamicArray = new float[3];

cout << myDynamicArray; // prints 8
cout << myDynamicArray[0]; no crash!

delete myDynamicArray;
```



```
float *myArray = new float[3];
```




```
char *myCharArray = (char*) myArray;
```



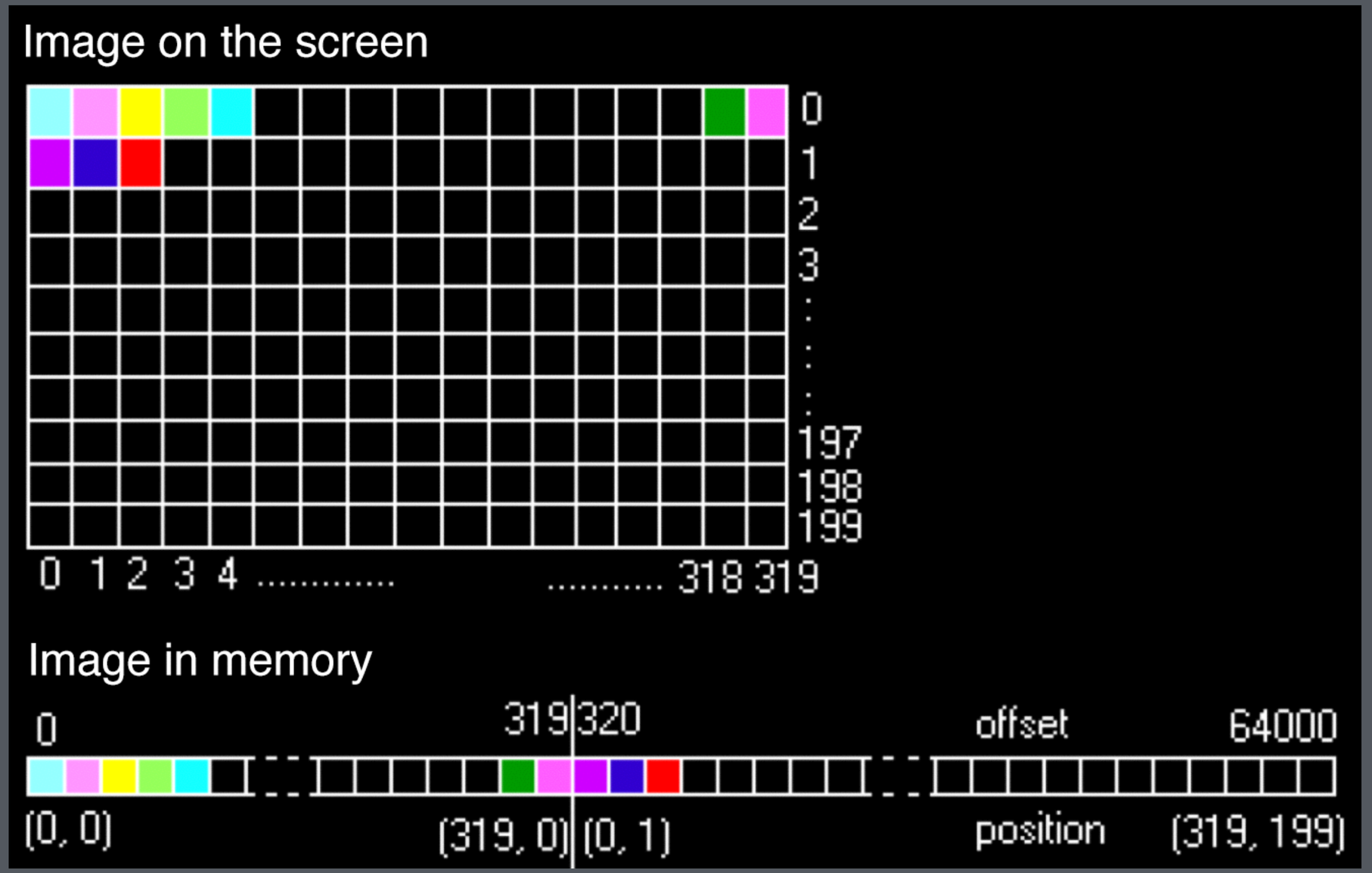
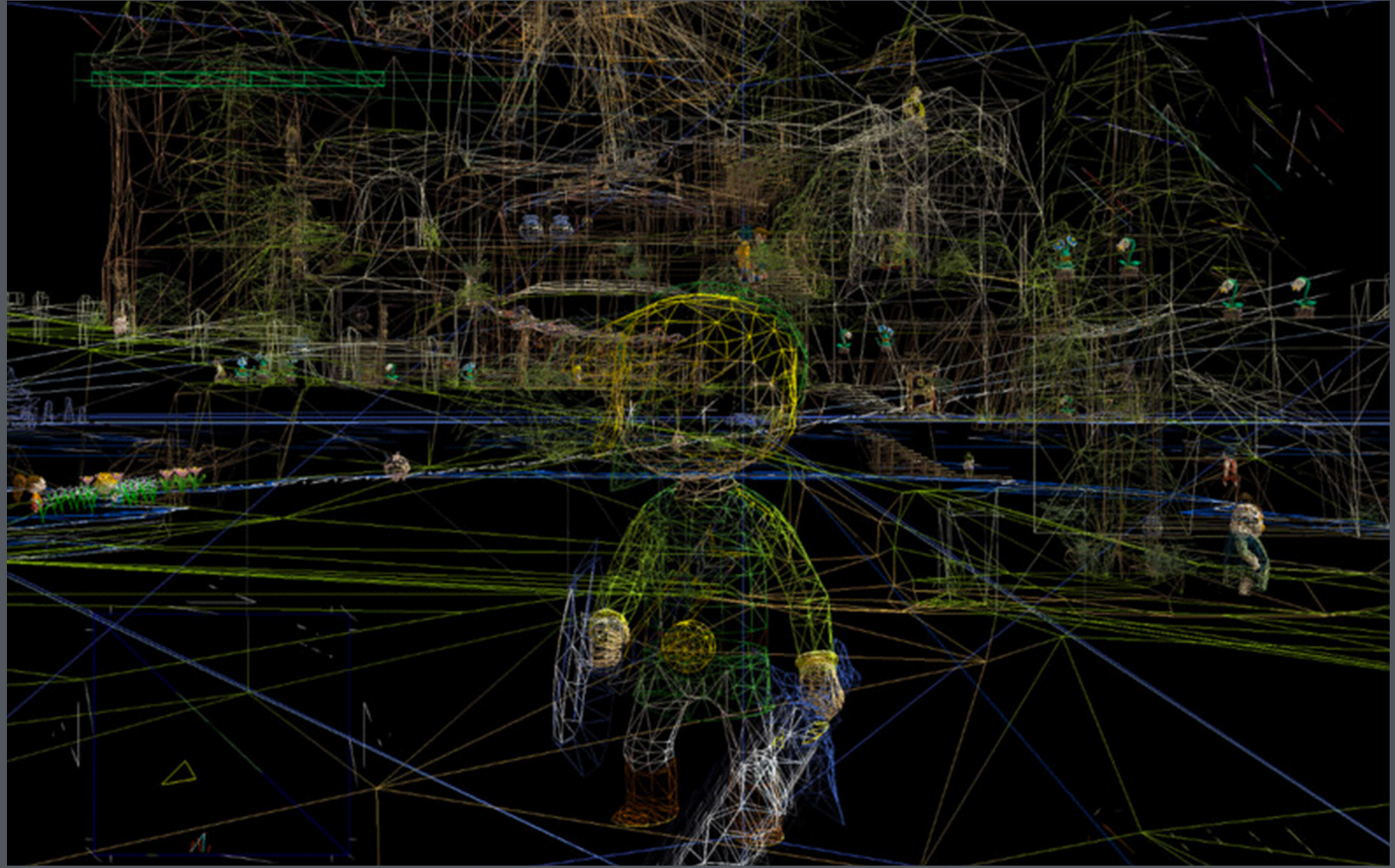
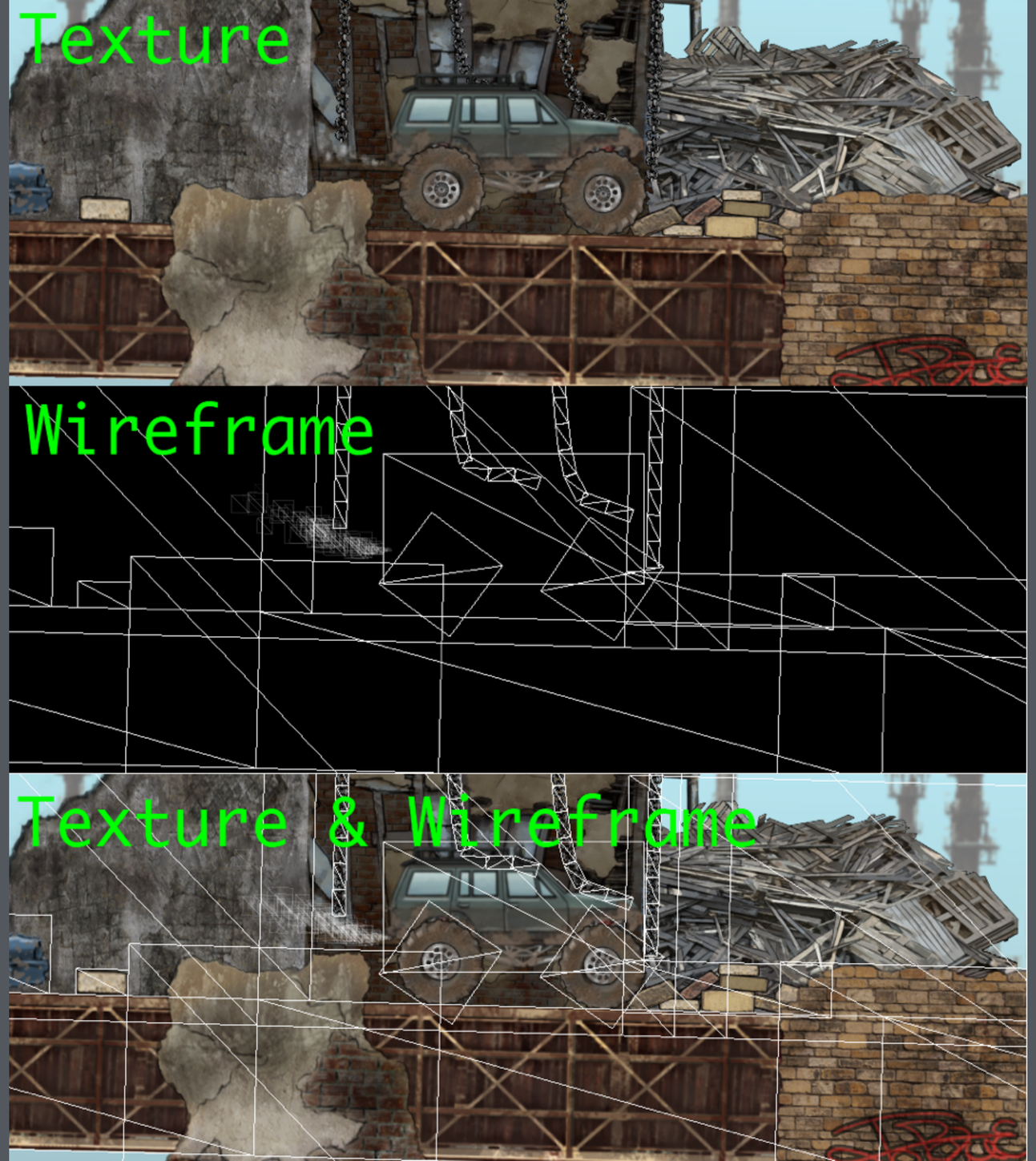


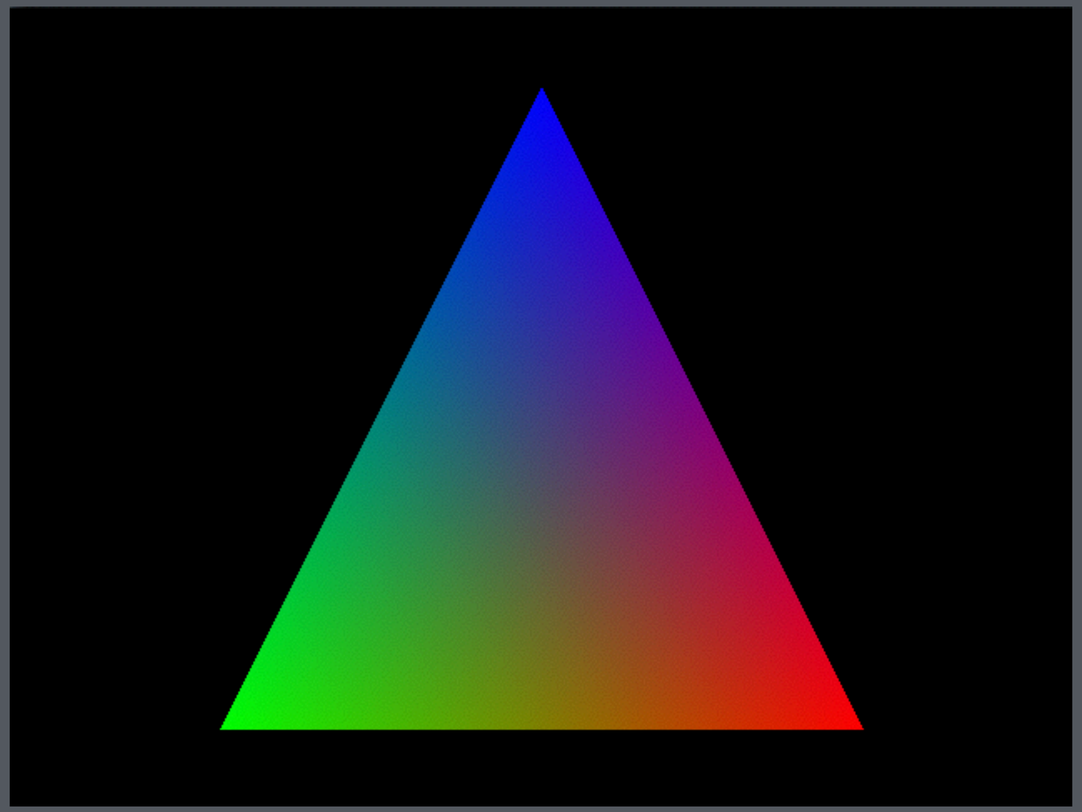
Image on the screen
0
1
2
3
197
198
199
0 1 2 3 4 ..... ..... 318 319
Image in memory
0
319
320
offset
64000
(0, 0)
(319, 0)
(0, 1)
position
(319, 199)

# **Graphics** in games

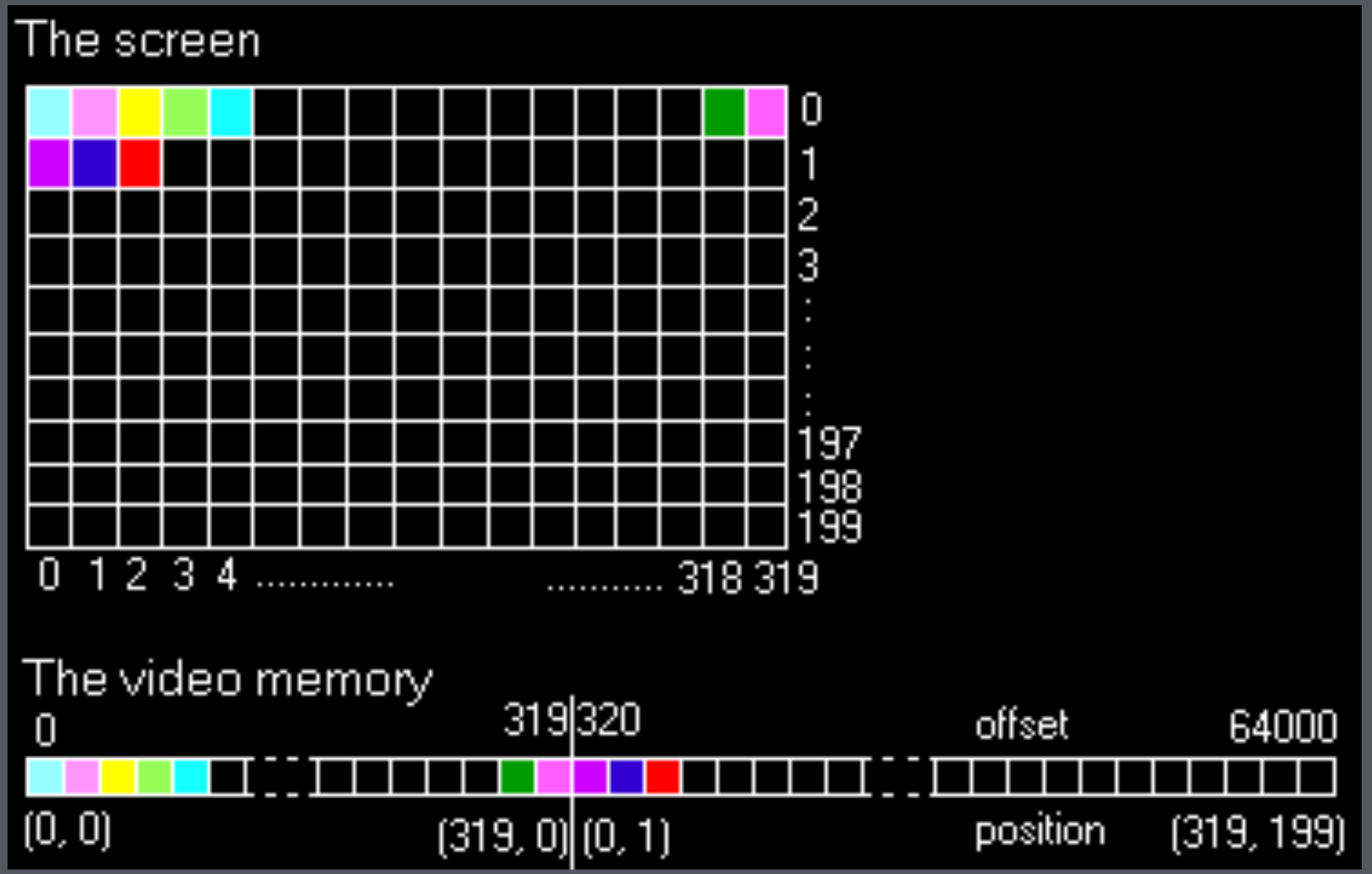

Defend
Put Away
72
FREE
4168



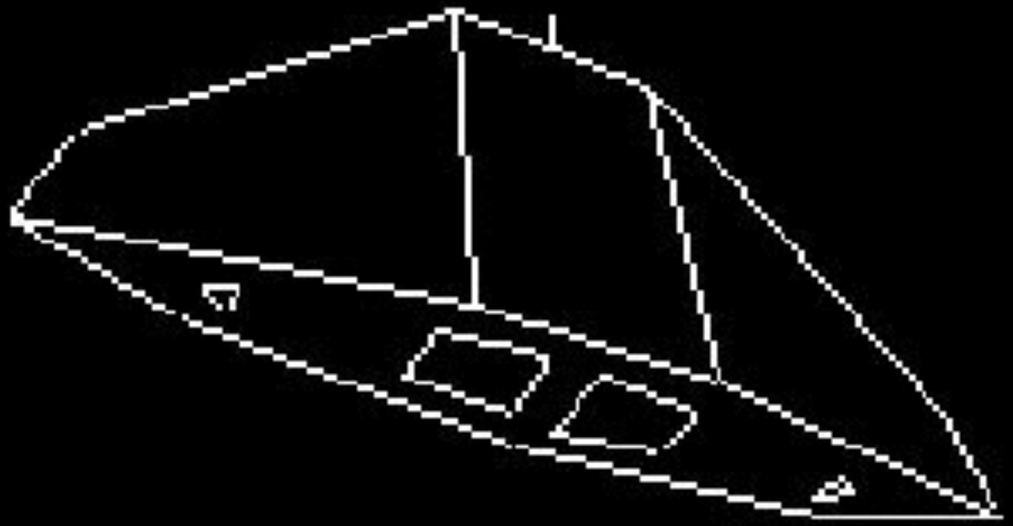

Texture
Wireframe
Texture & Wireframe



Why **triangles**?


The screen
0
1
2
3
197
198
199
0 1 2 3 4 ..... ..... 318 319
The video memory
0
319
320
offset
64000
(0, 0)
(319, 0)
(0, 1)
position
(319, 199)

# **Software** rendering



Load New Commander (Y/N)?

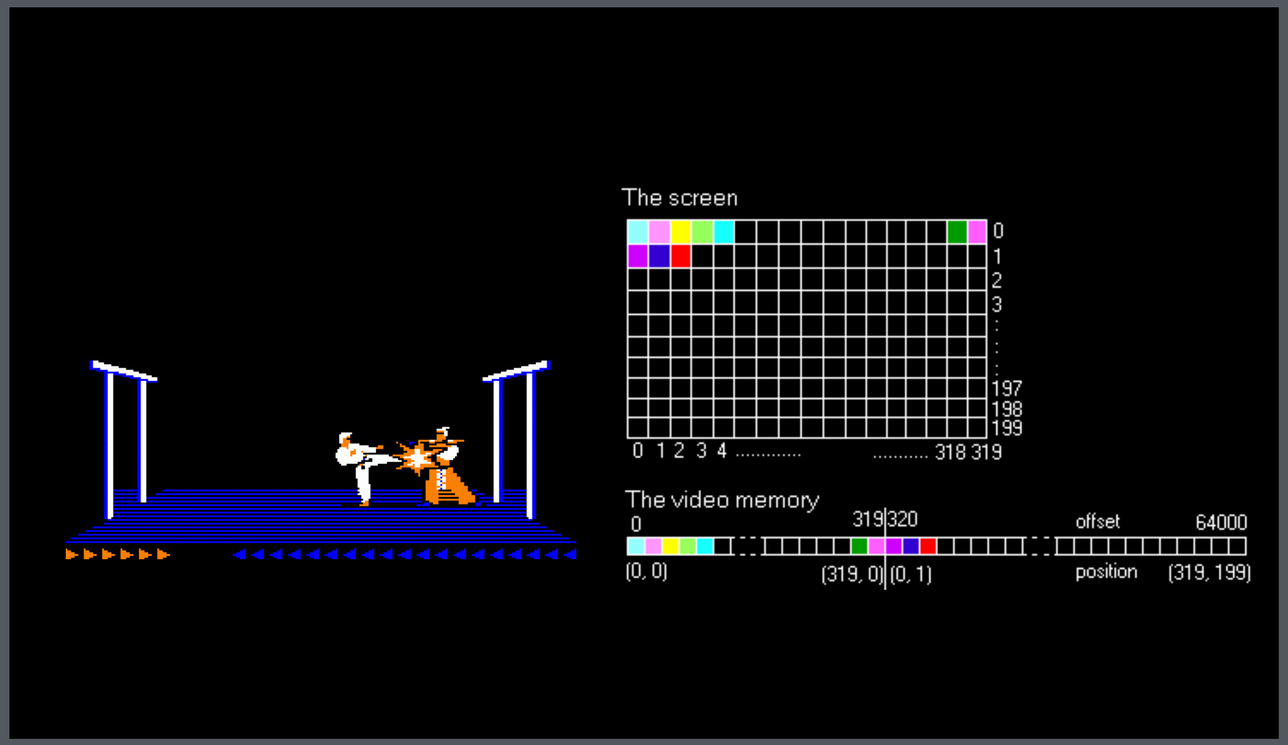

SPD: 0
VSI: 01.78
THR:90%
HDG:0
ALT:83


93
75
13
0
78
91
13

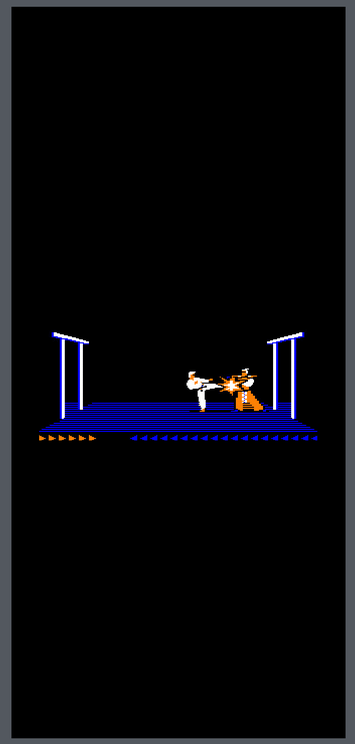

LEVEL UP
CHAR
QUESTS
MAP
MENU
INV
SPELLS
RED STORM
TOTAL KILLS : 19
RESISTS : LIGHTNING
IMMUNE : MAGIC

# Software Rendering

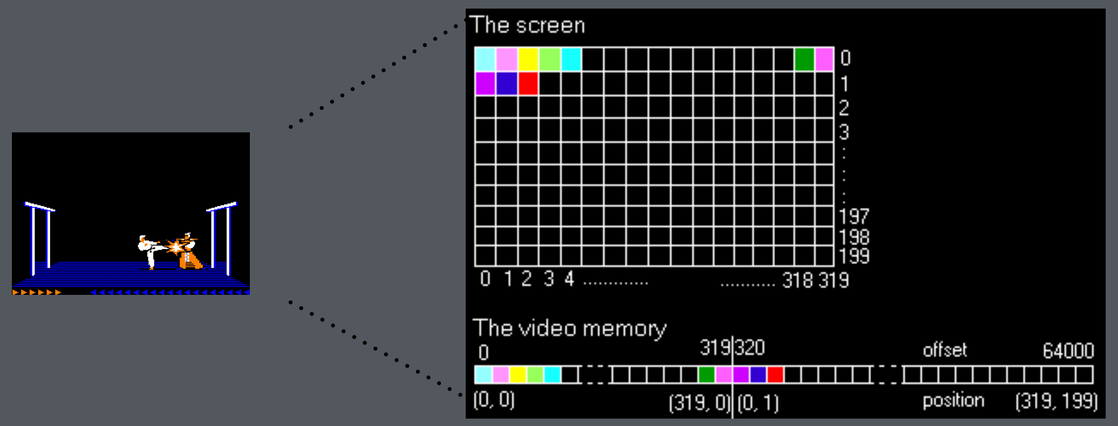
## RAM




## Video card



# Software Rendering




2560 * 1600 = **4,096,000** pixels

~12 MB == 4 * sizeof(**War and Peace**)

**60 times a second**

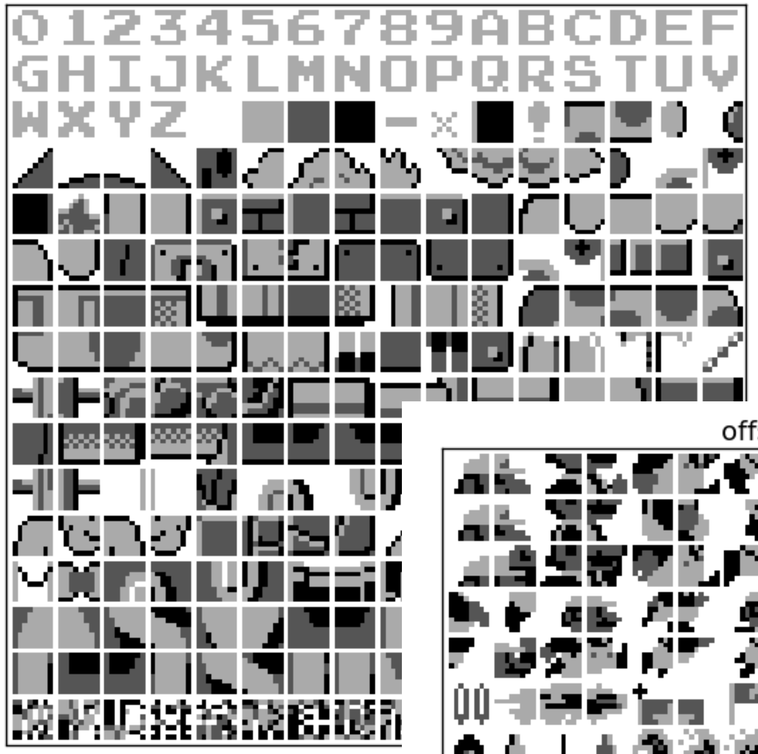
# **Hardware** rendering


Galaga
DOUBLE DONKEY KONG
Nintendo
MIDWAY
Bally
PAC-MAN
Home Theater of Arcades
PLEASE DO NOT START GAMES
25¢
25¢


Nintendo
ENTERTAINMENT SYSTEM
POWER
RESET
SELECT START
Nintendo
B
A

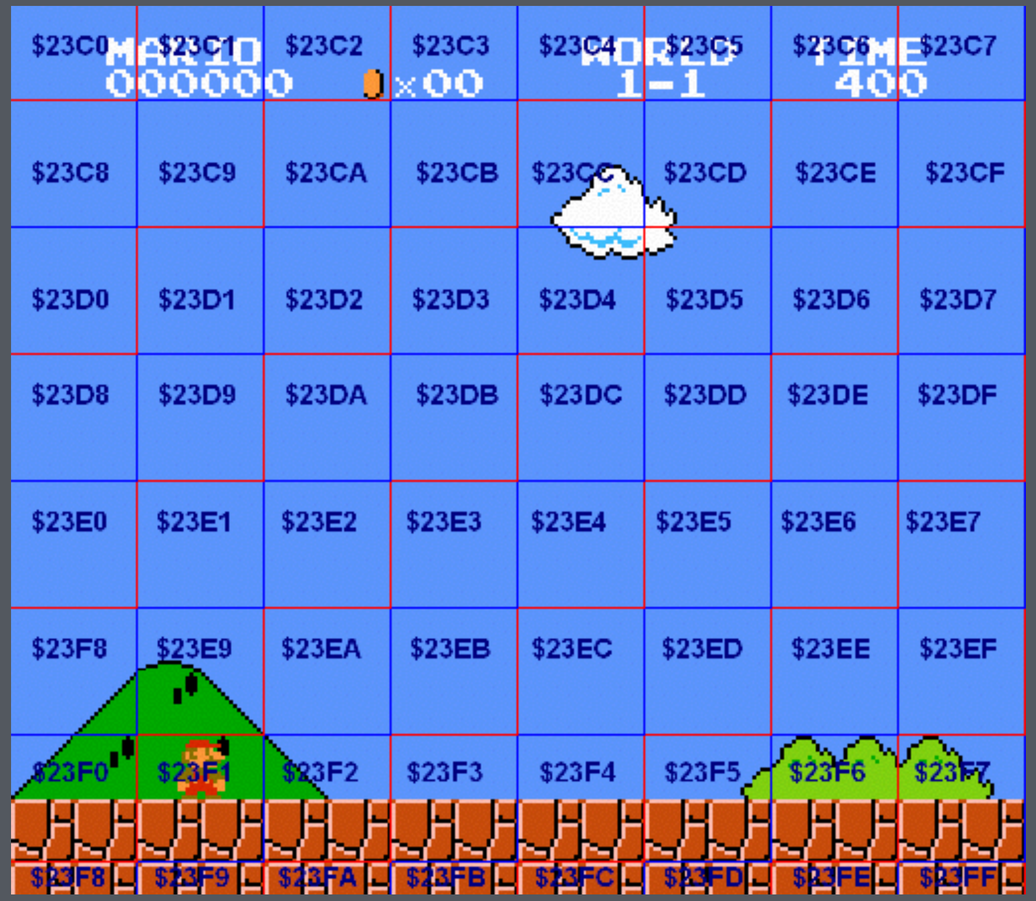

MARIO
000000
×00
WORLD
1-1
TIME
SUPER
MARIO BROS.
©1985 NINTENDO
1 PLAYER GAME
2 PLAYER GAME
TOP- 000000

offset = 2305



offset = 2049

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |

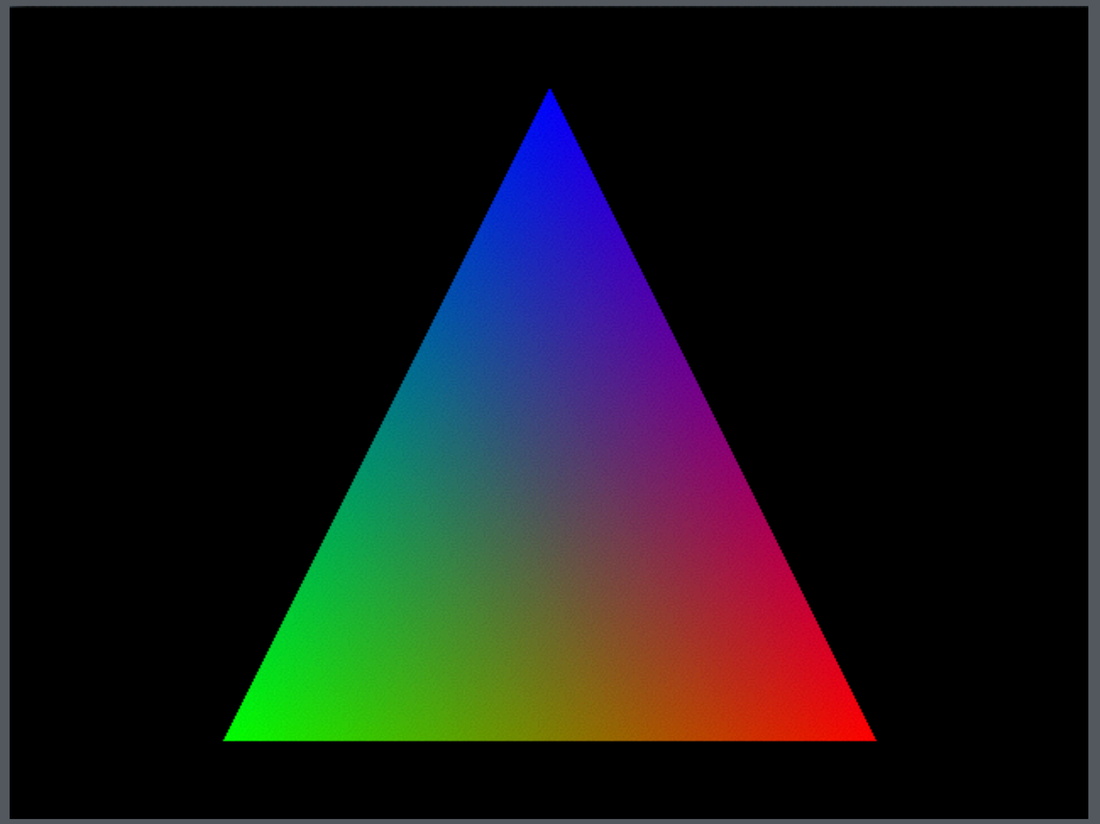



DAYTONA
USA
RACE LEADER
RACE LEA
50
2ND/2
50
1ST/2

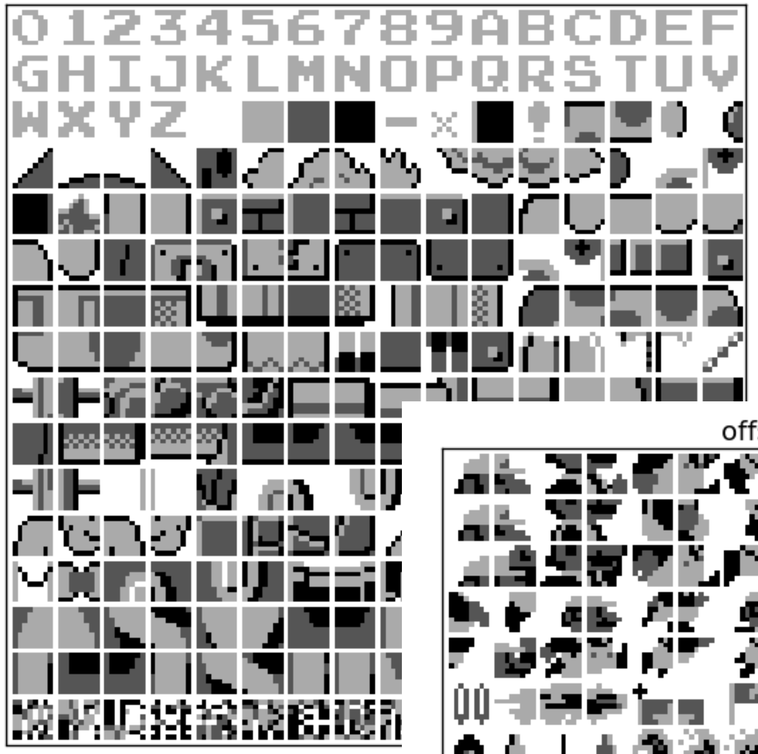

12MB
BLASTER
Hardware XTRA AWARD
PC-Games 05/98
Hardware XTRA AWARD
PC-Action 05/98
PC Player
PC-Player 06/98
Voodoo 2
FASZINIERENDE SPIELE
4
FASZINIERENDE SPIELE
INCOMING
actua SOCCER 2
ULTIMATE RACE PRO
3Dfx
Get the Magic of Speed!
Basiert auf dem neuen Voodoo2 Graphics™ Chipsatz von 3Dfx Interactive™
Ausgestattet mit 12 MB Hochleistungsspeicher für stärkste Leistung
Bis zu 50 Milliarden Operationen und 3 Millionen Triangles pro Sekunde
Bis zu dreimal schnellere 3D-Verarbeitung als beim ursprünglichen Voodoo Graphics™-Chipsatz!
Arbeitet mit Ihrer vorhandenen Grafikkarte zusammen und bietet Ihnen das schnellste 3D-Spiel aller Zeiten
Enthält 4 topaktuelle Spiele
CREATIVE
WWW.SOUNDBLASTER.COM


11
108
20
0
0


PARK
ONE W

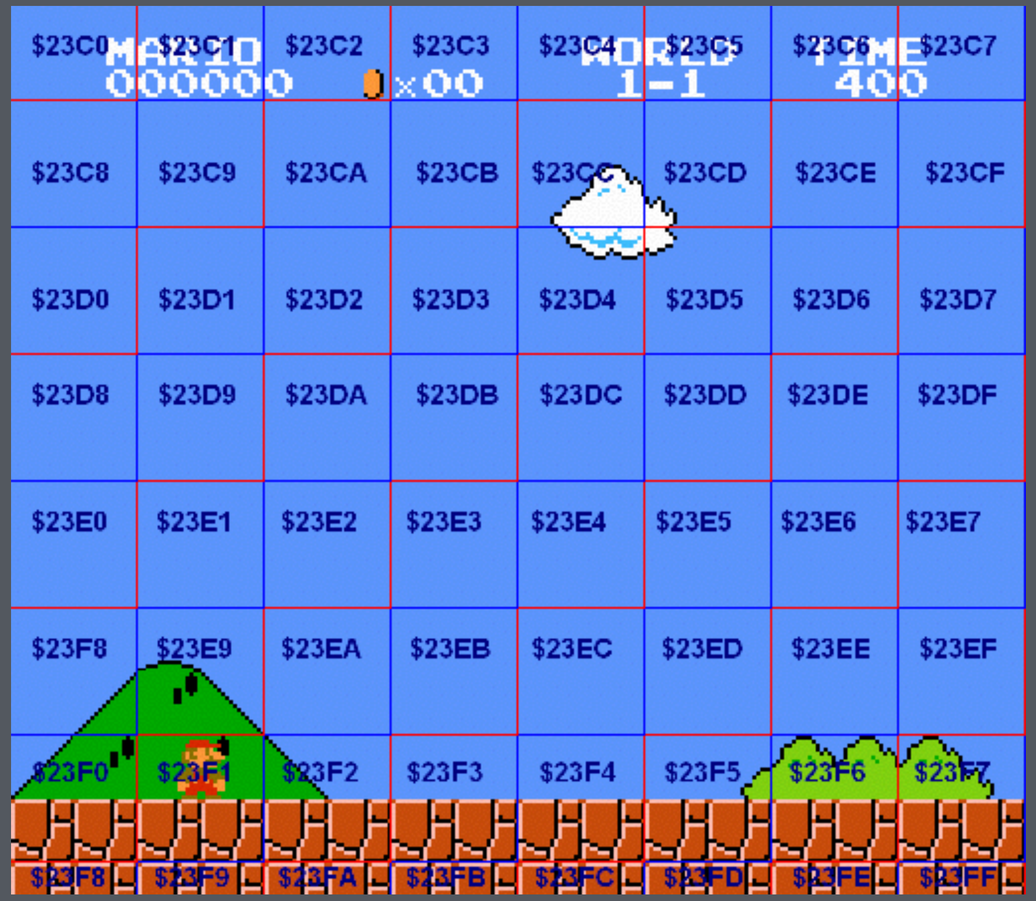


offset = 2305

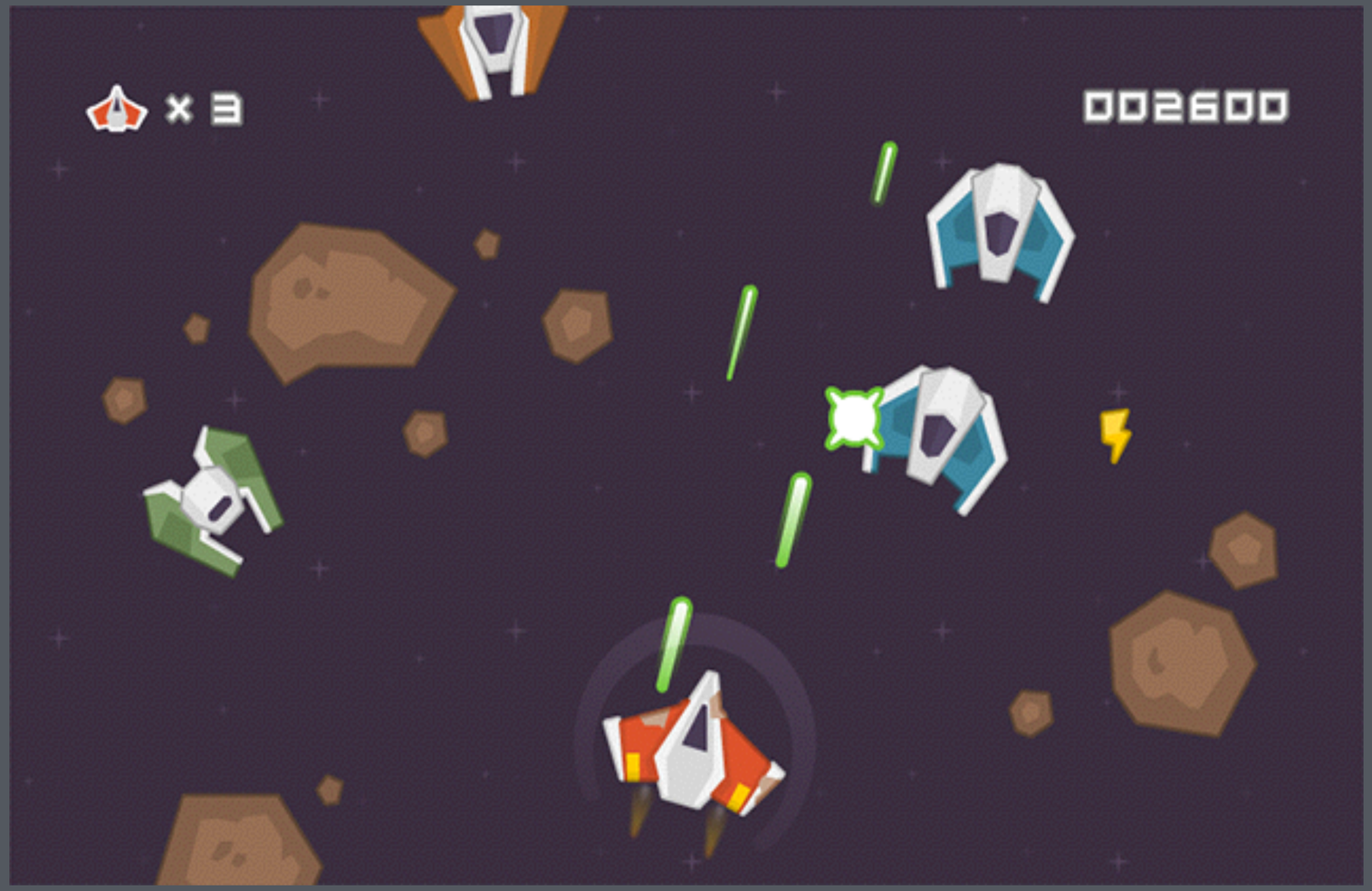


offset = 2049

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |

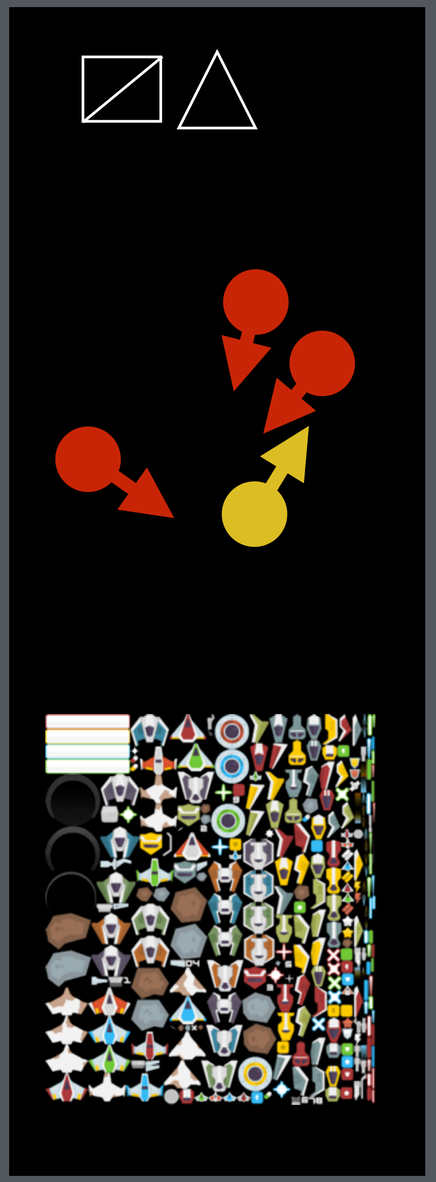


# Hardware Rendering

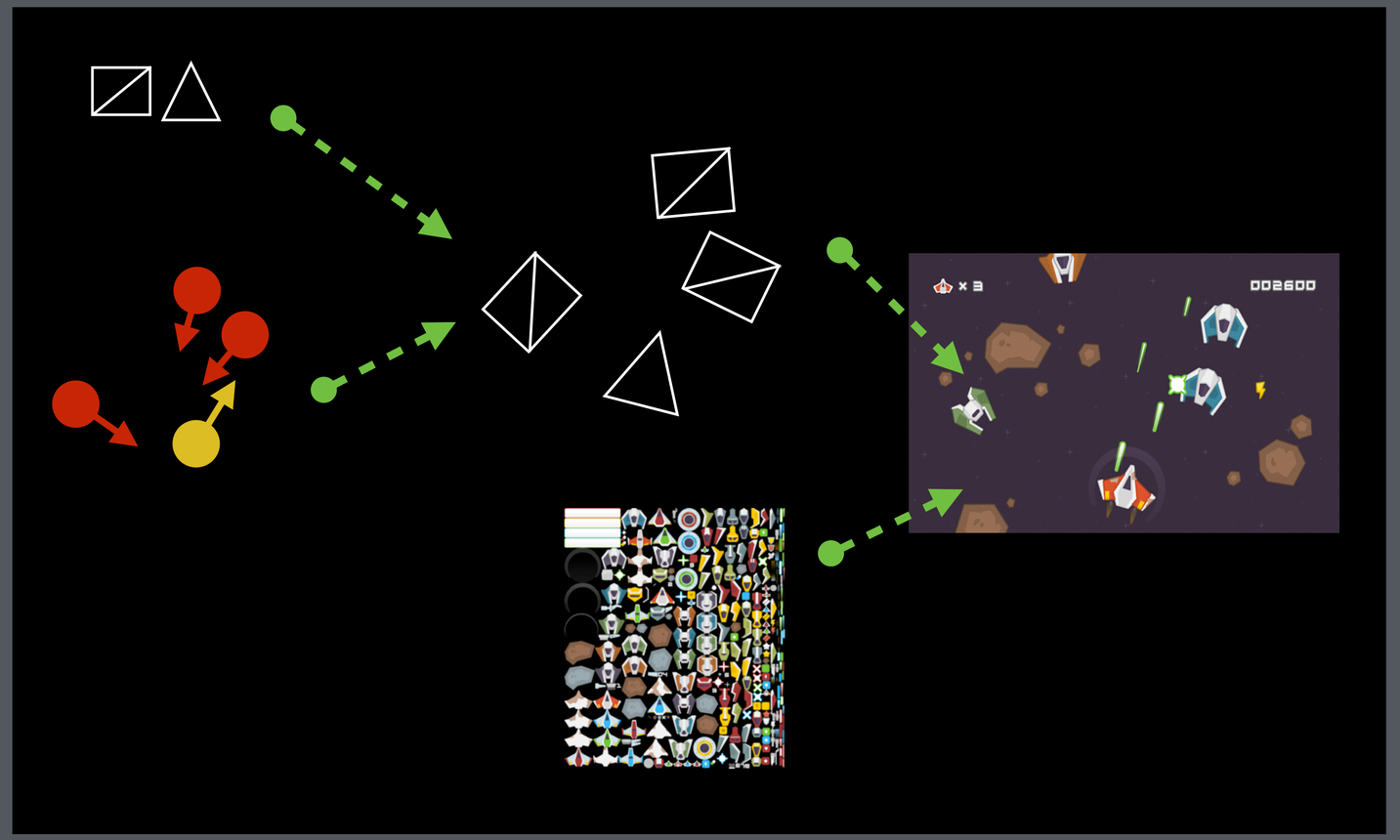


# Hardware Rendering

RAM



Video card



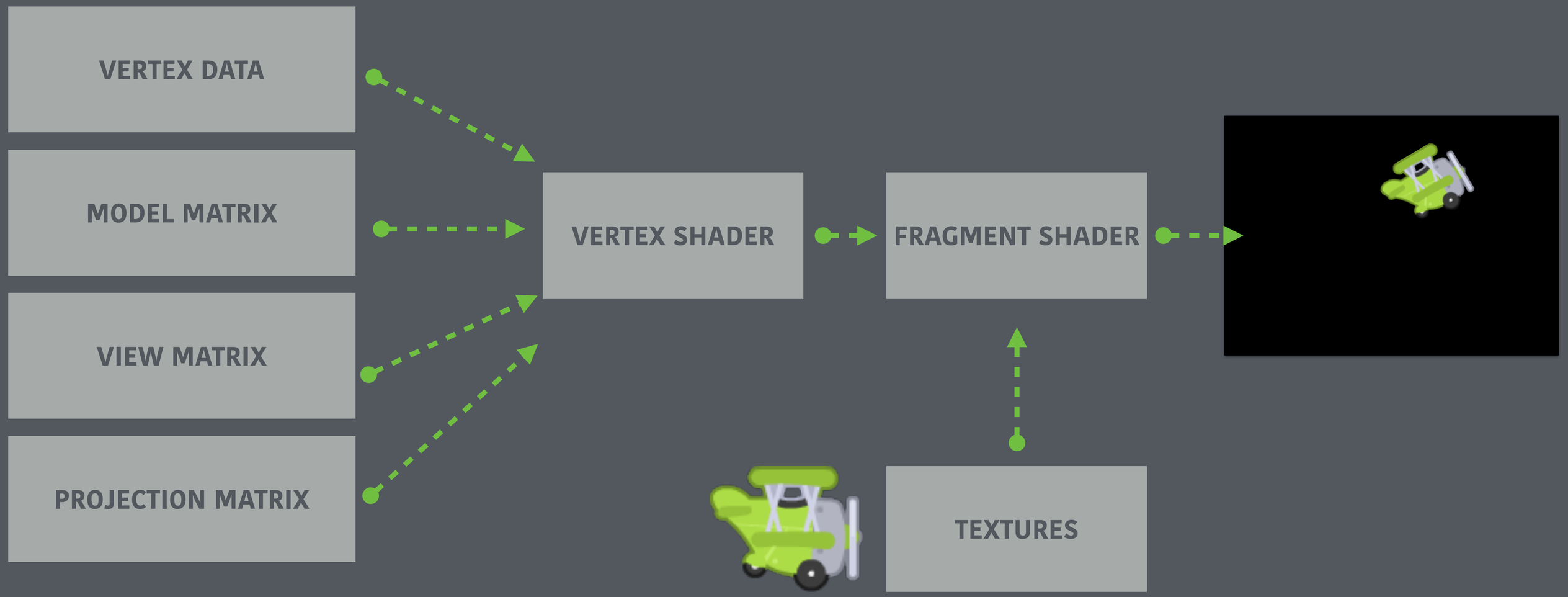

# The **GPU** pipeline

**What?**

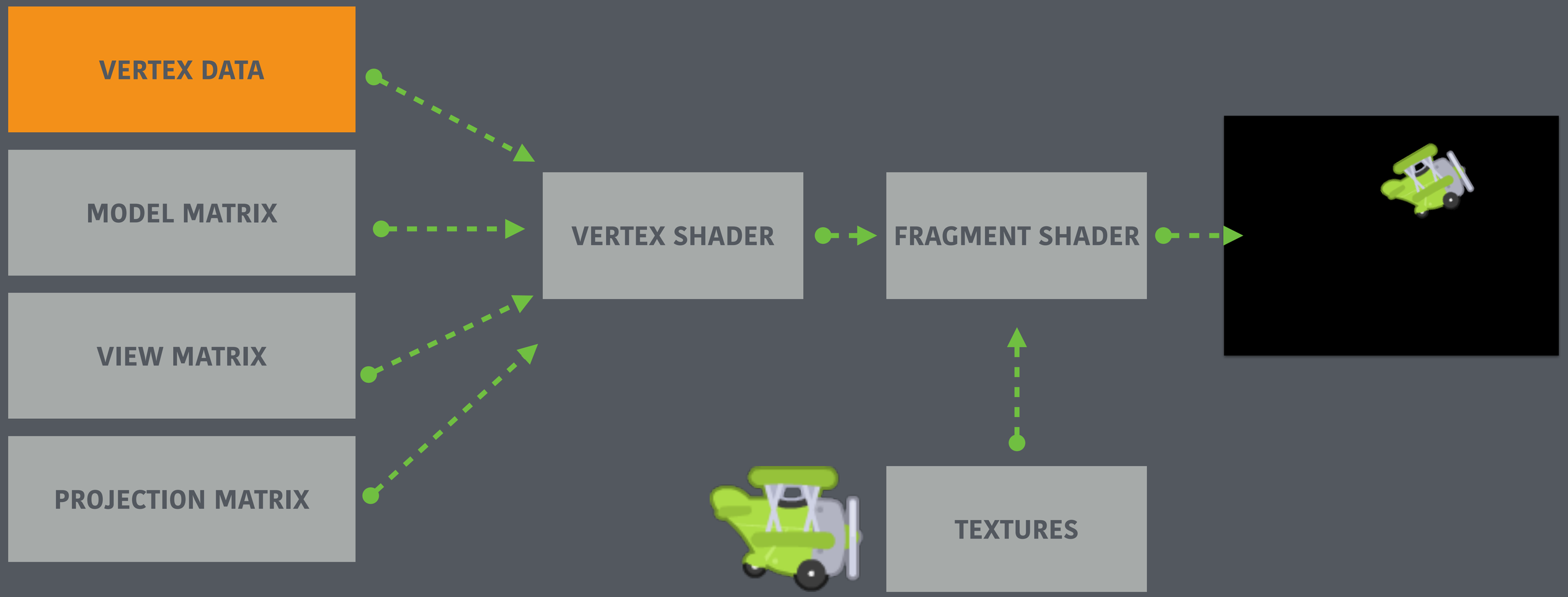
Where?

**How?**

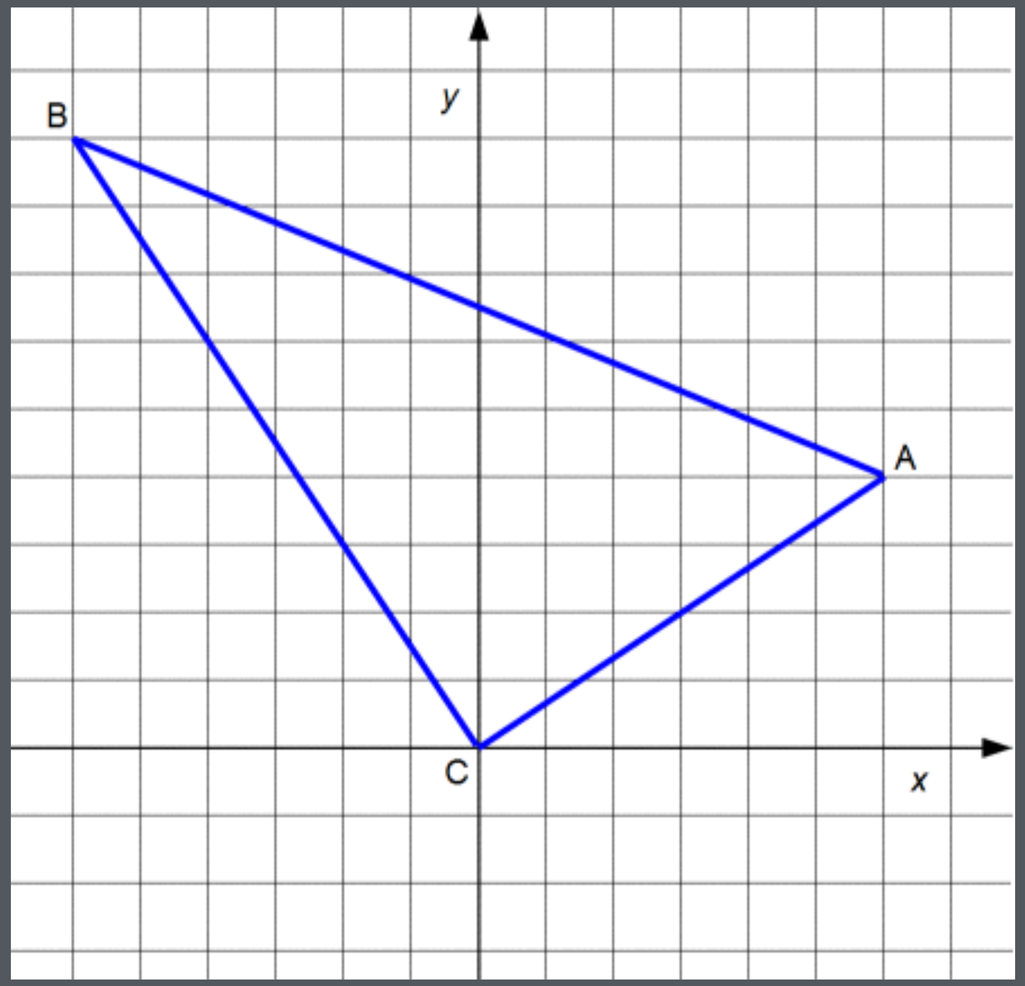
# The **GPU** pipeline



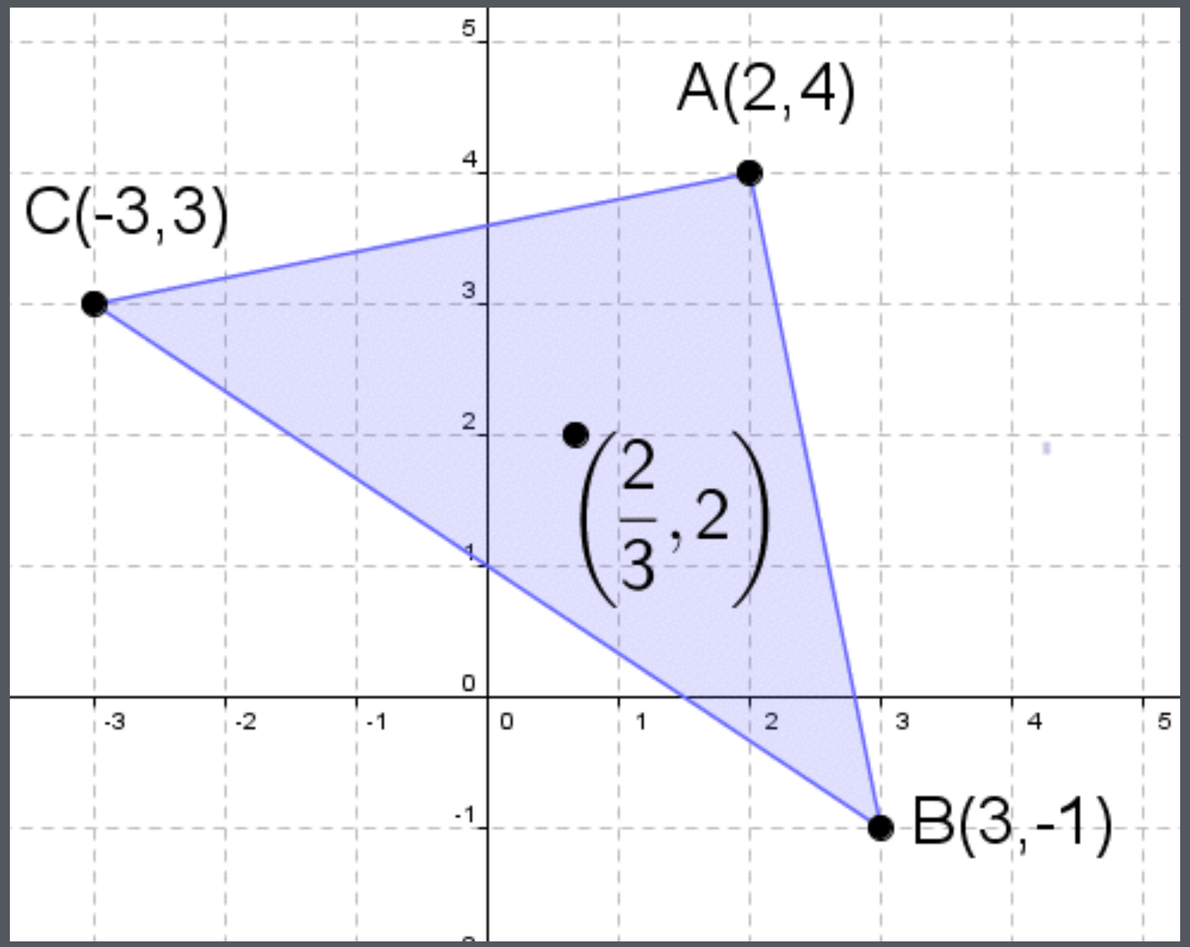
# The **GPU** pipeline

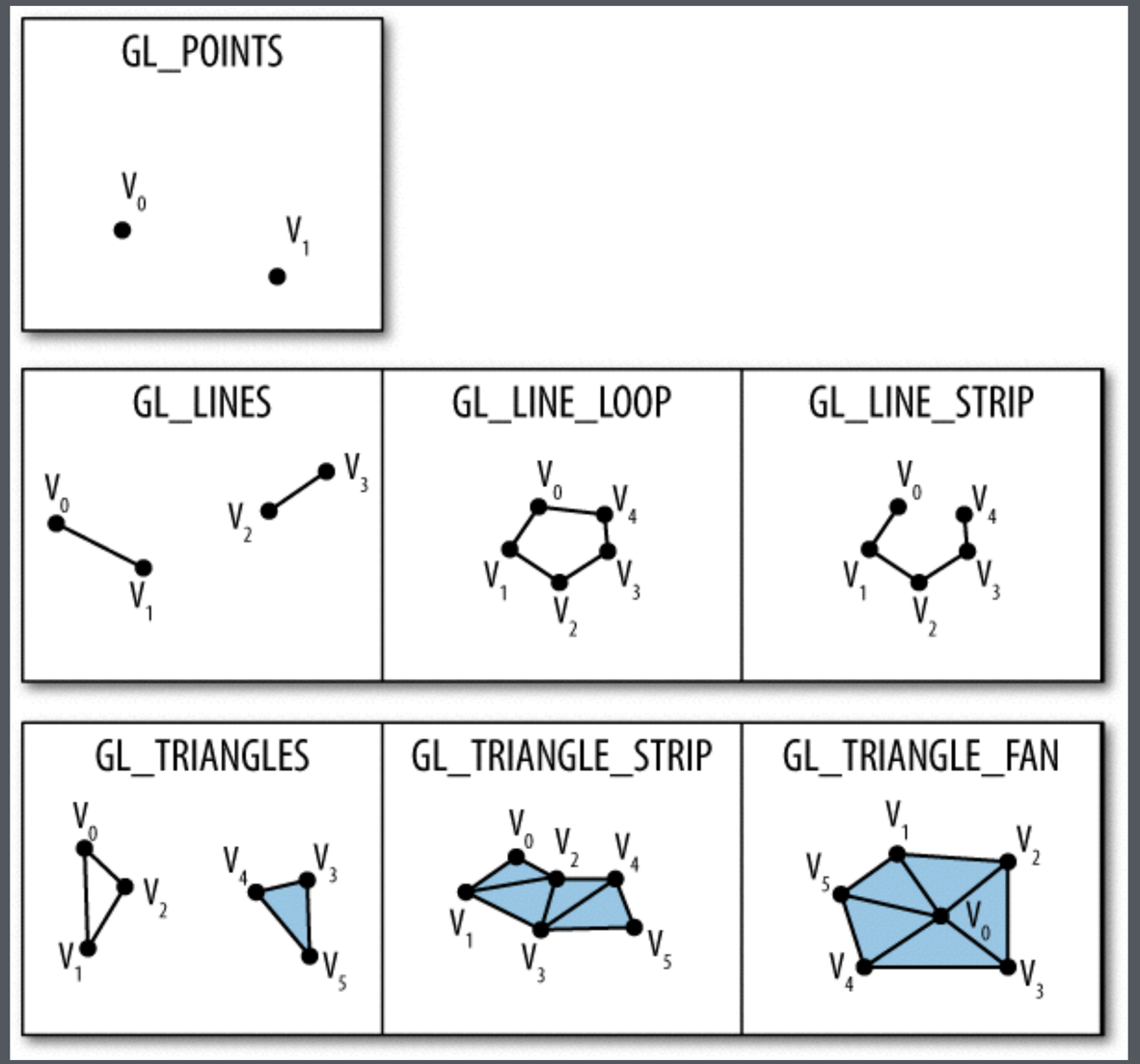


# **Vertex** data

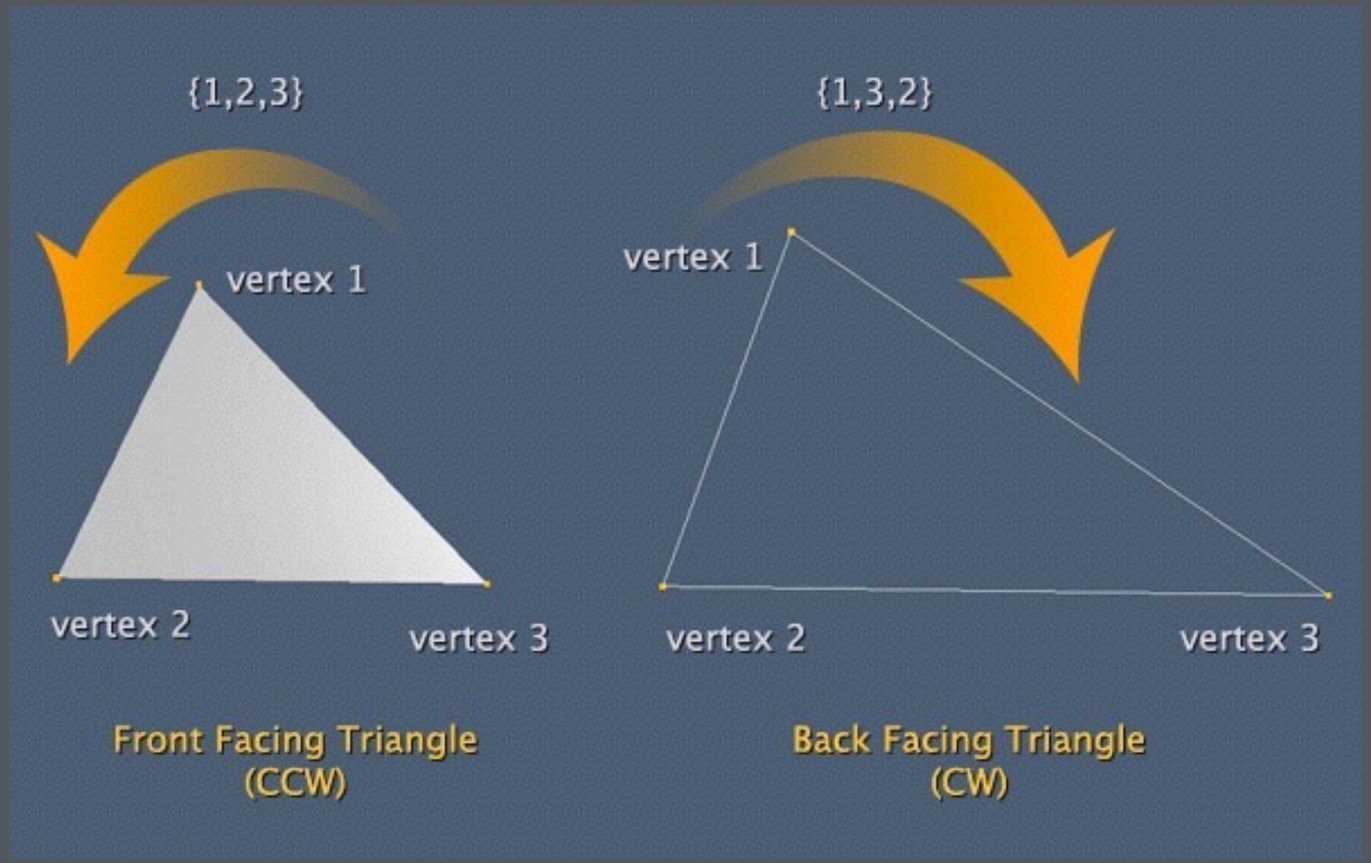

B
y
A
C
x

A **polygon** is defined by
points in space called **vertices**

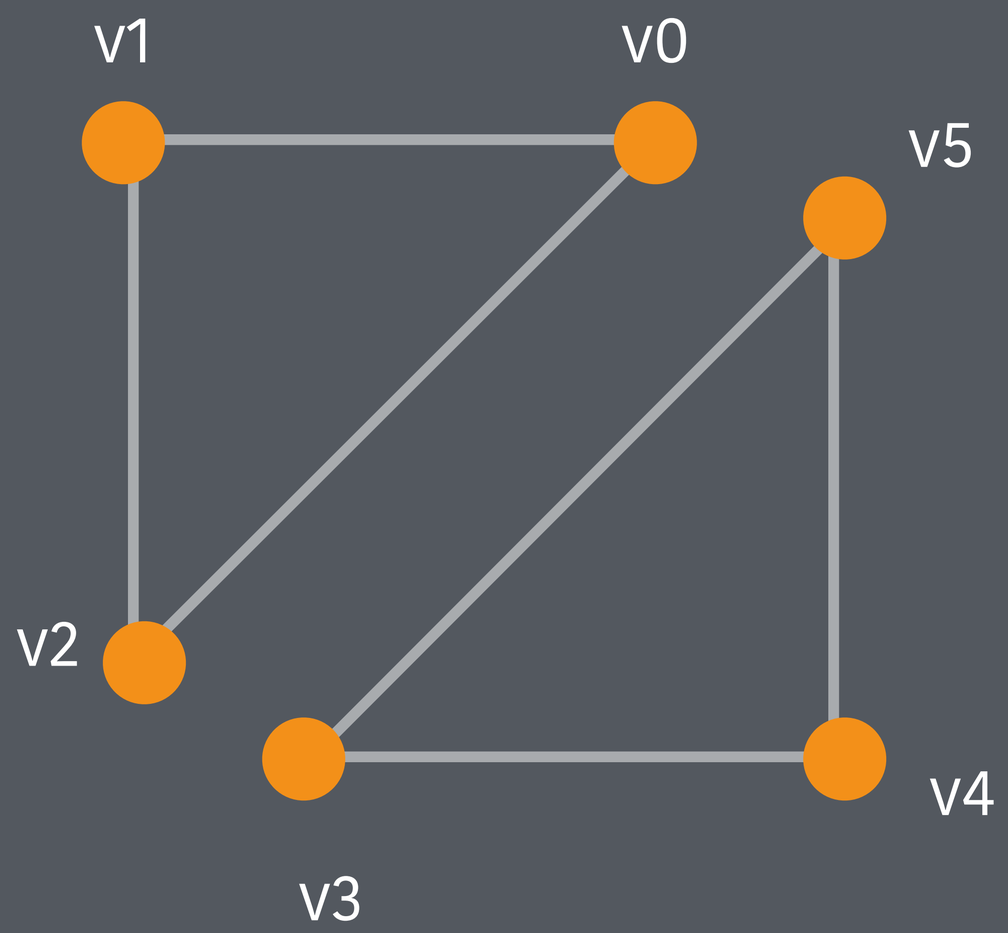

A(2,4)
C(-3,3)
$\left(\frac{2}{3},2\right)$
B(3,-1)
5
4
3
2
1
0
-1
-3
-2
-1
0
1
2
3
4
5

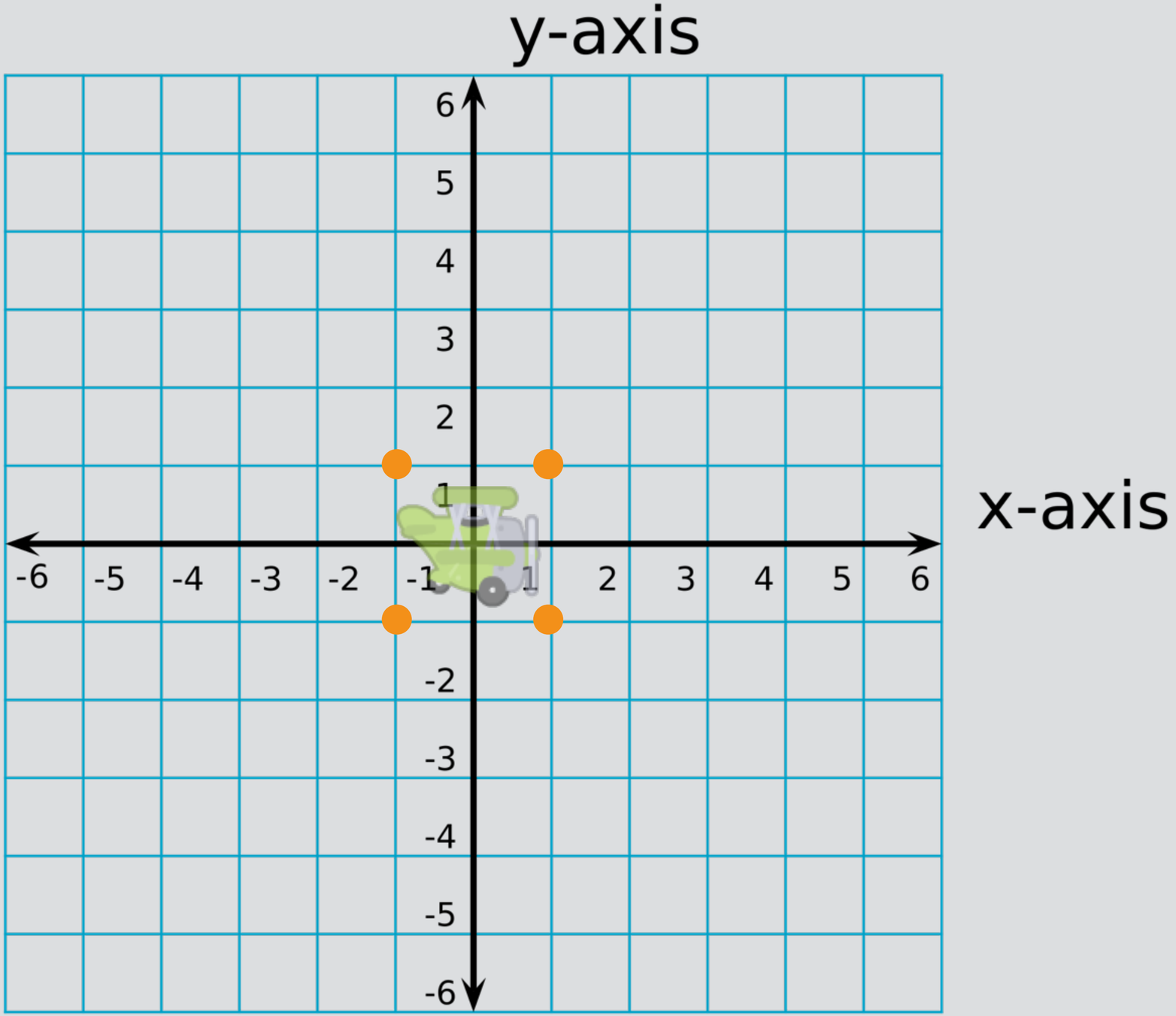

GL_POINTS
V0
V1
GL_LINES
V0
V1
V2
V3
GL_LINE_LOOP
V0
V1
V2
V3
V4
GL_LINE_STRIP
V0
V1
V2
V3
V4
GL_TRIANGLES
V0
V1
V2
V3
V4
V5
GL_TRIANGLE_STRIP
V0
V1
V2
V3
V4
V5
GL_TRIANGLE_FAN
V0
V1
V2
V3
V4
V5

Polygons are **one-sided** and the side is defined by the **order of the vertices**

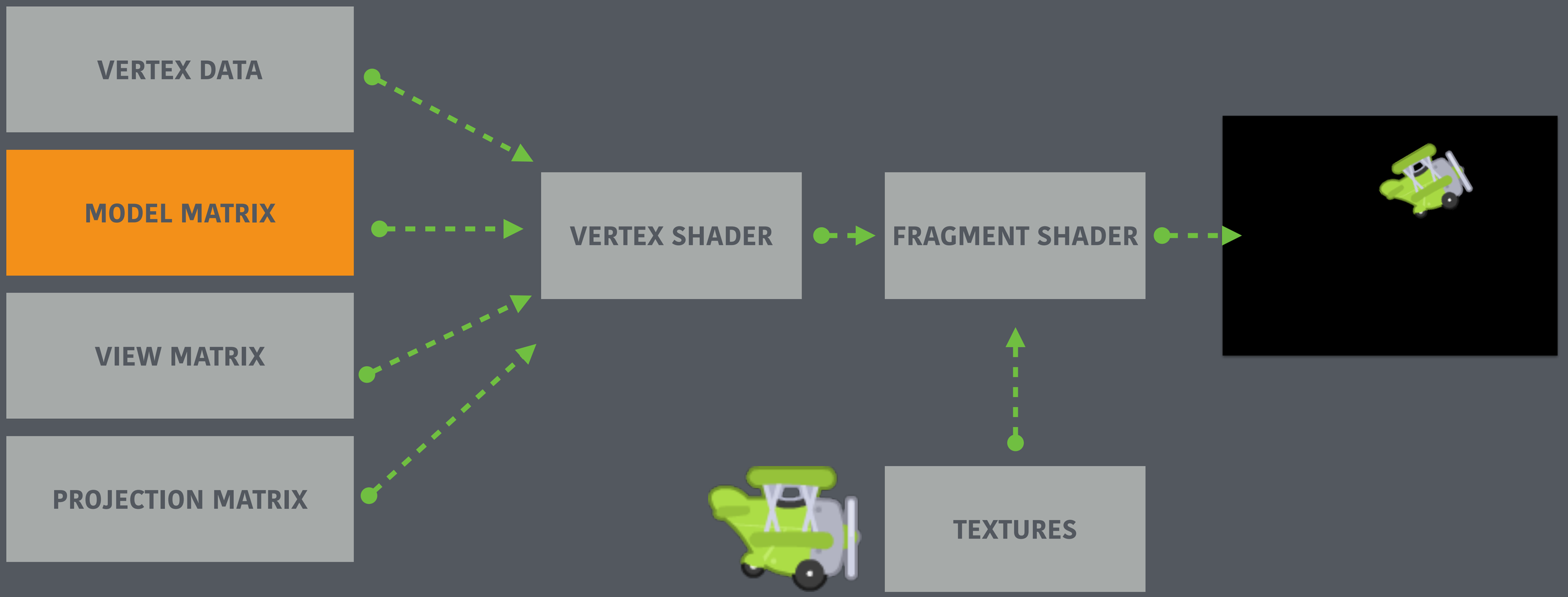


**Counter-clockwise** order!


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

# The **GPU** pipeline



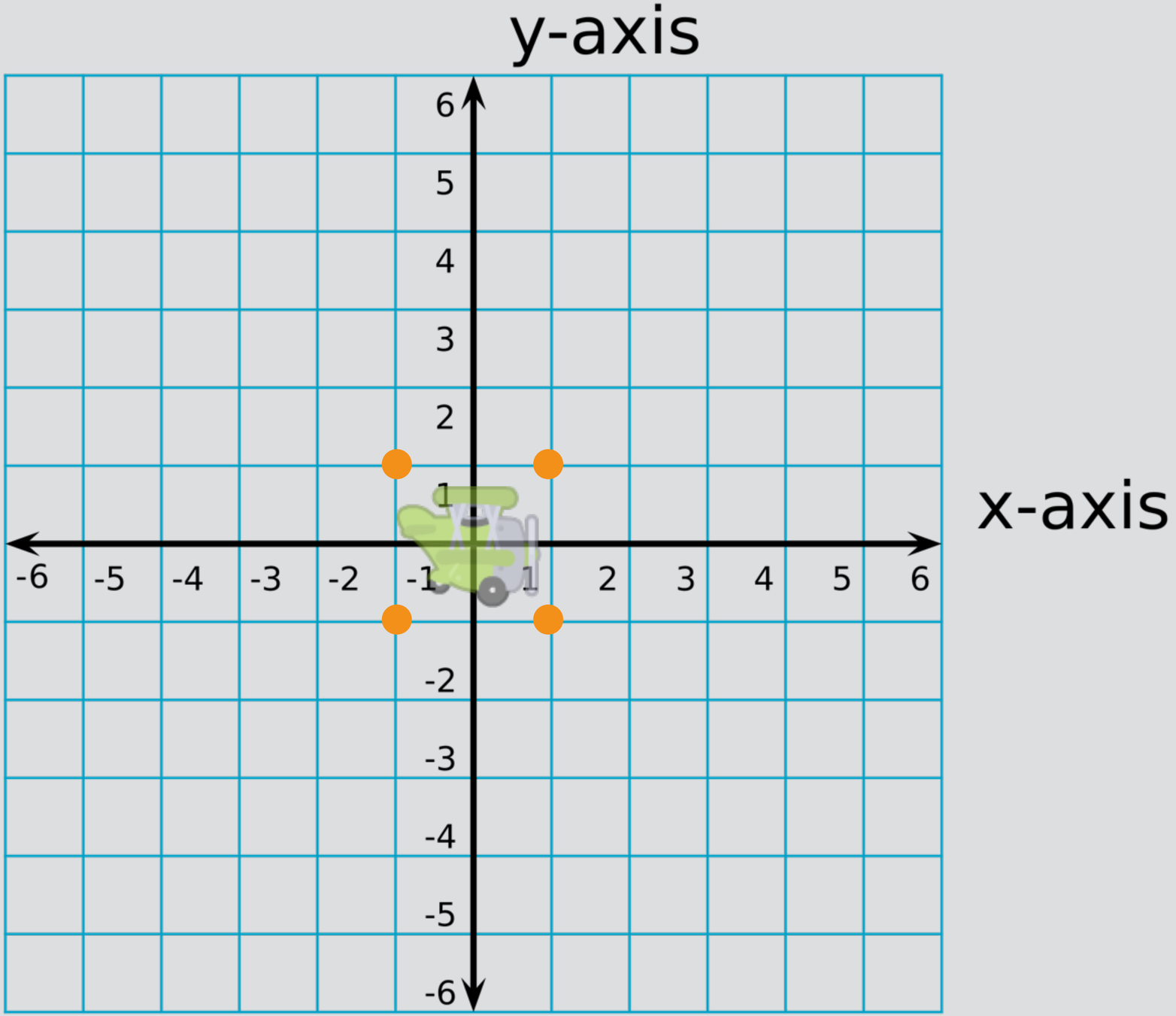
# The **model** matrix

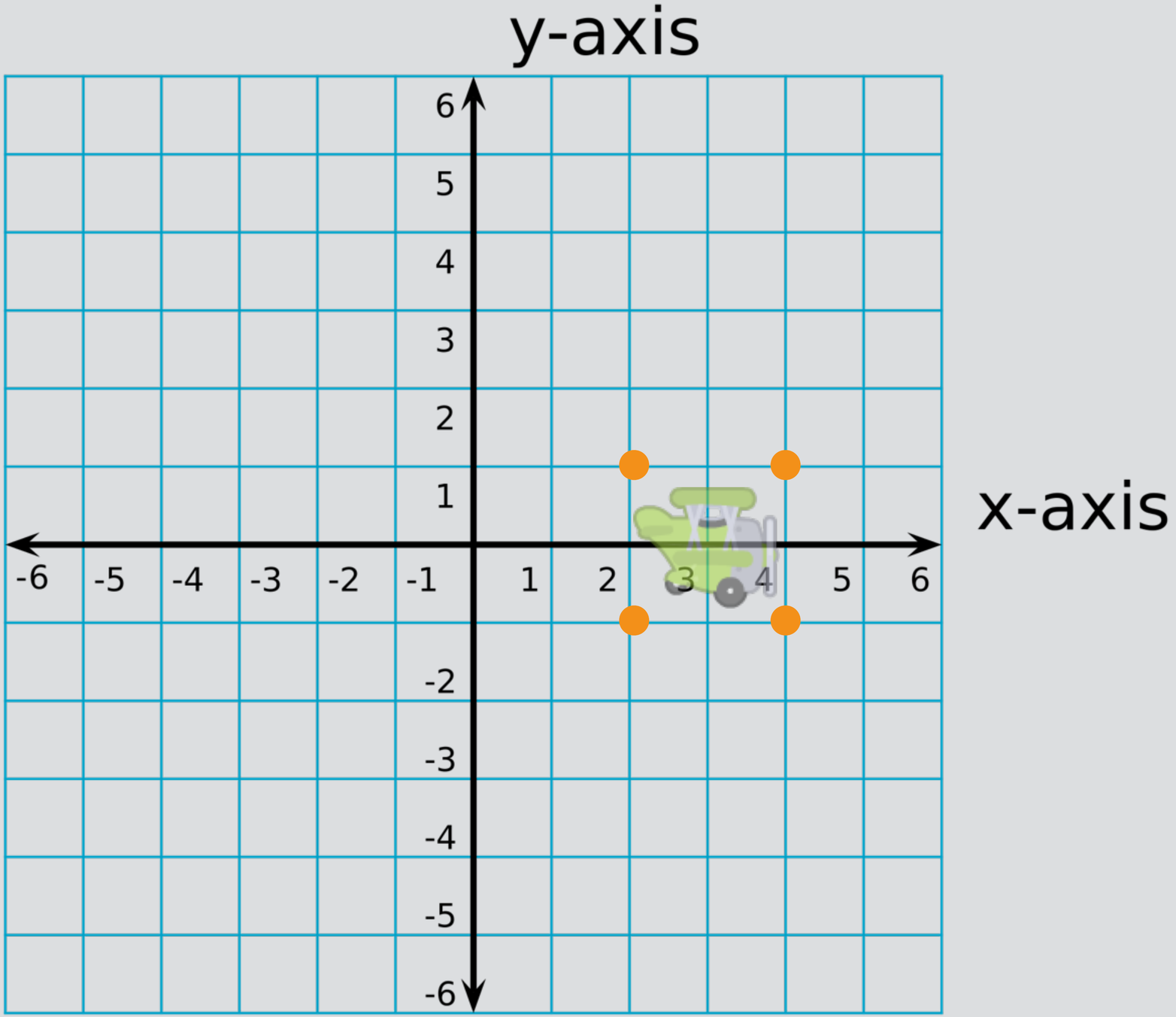
# What is a **matrix**?

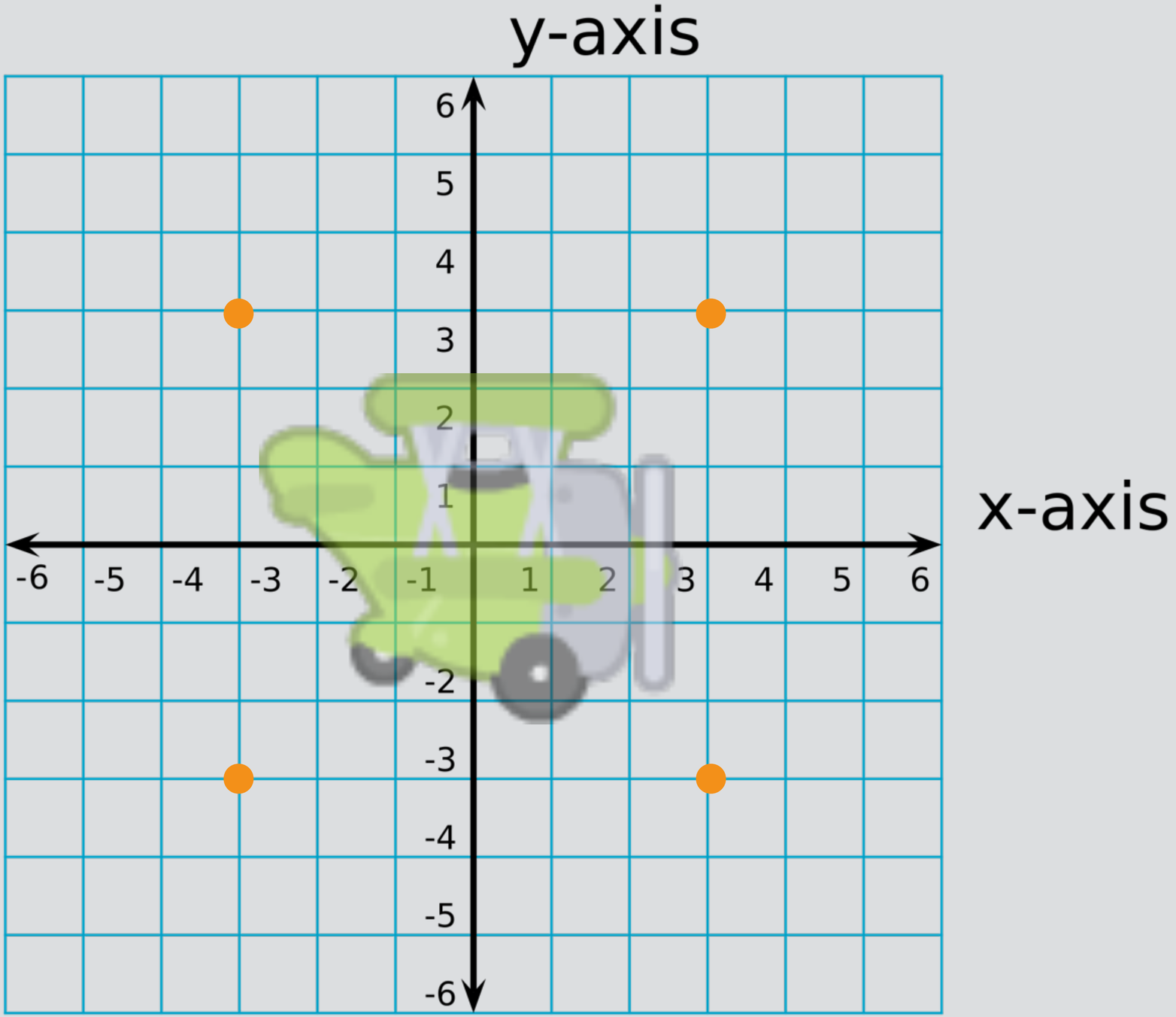
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

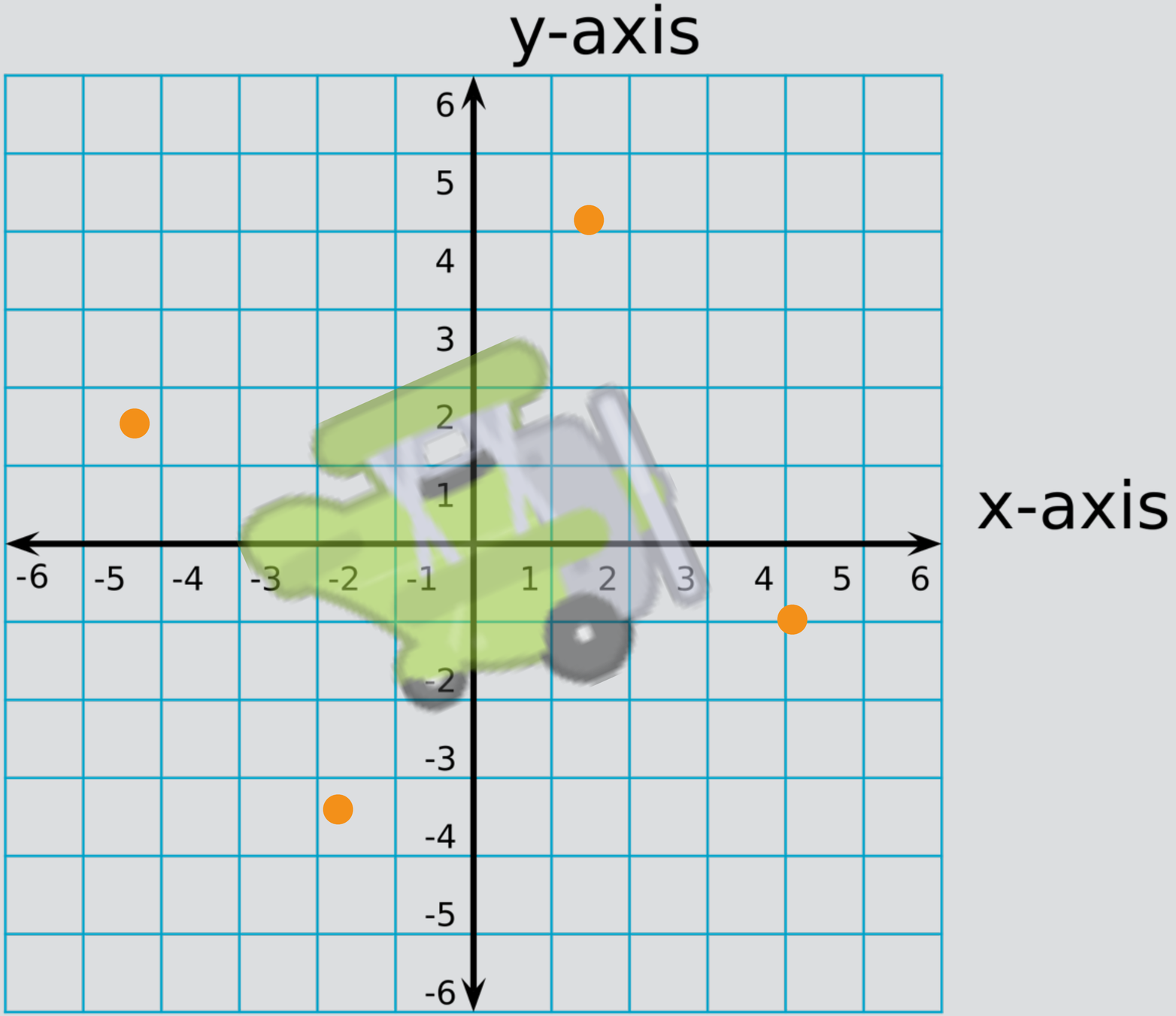
# **Transformation** matrix

Represents a linear transformation in space
(**move**, **rotate**, **scale**).


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

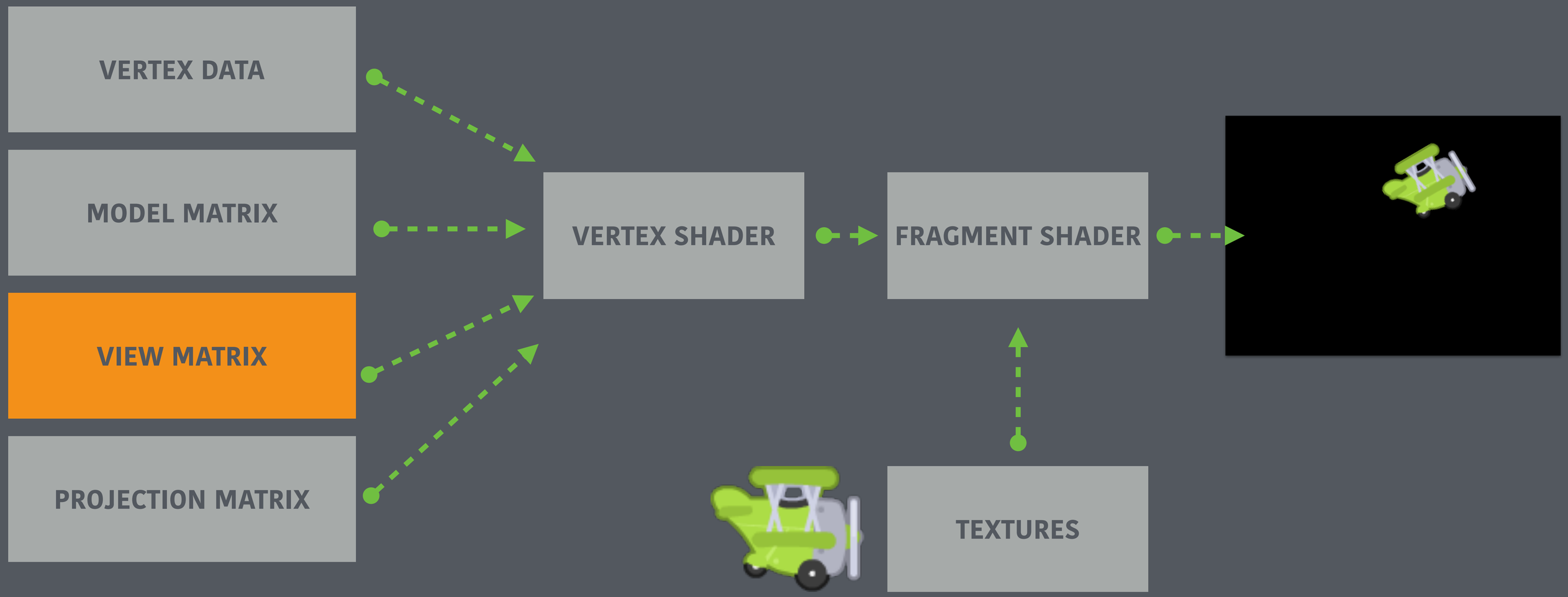

y-axis
x-axis
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6


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


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

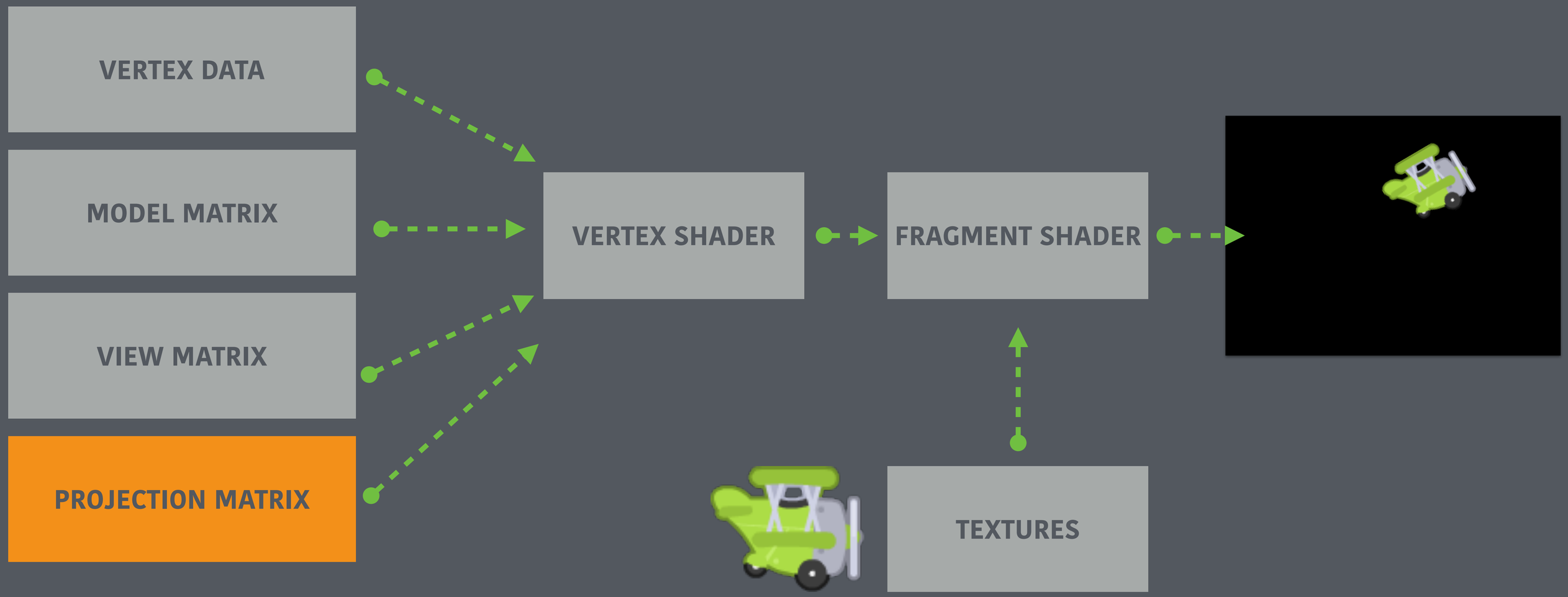
The **model** matrix is a **TRANSFORMATION MATRIX**
that defines the **transform** of each object you draw

# The **GPU** pipeline



# **View** matrix

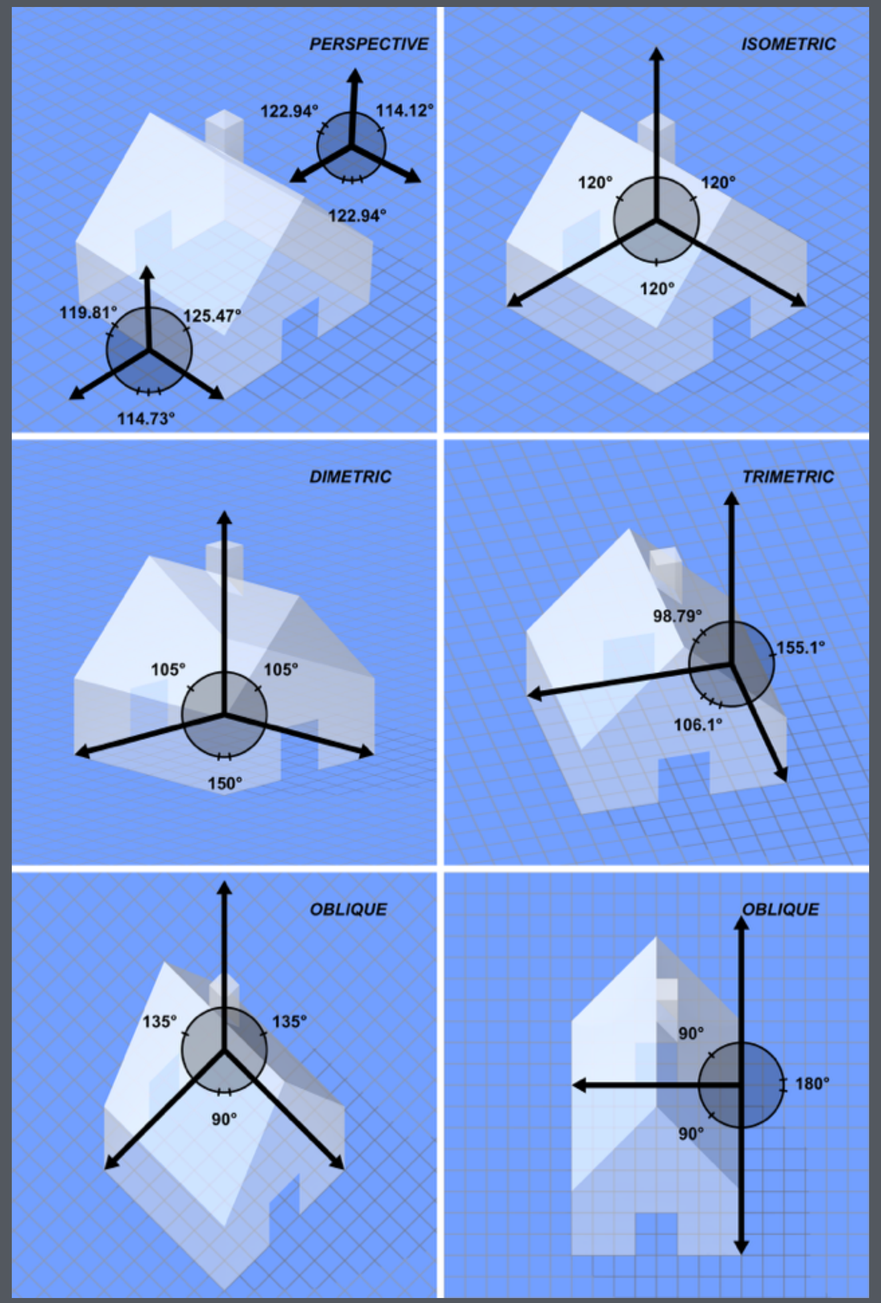
# The **GPU** pipeline



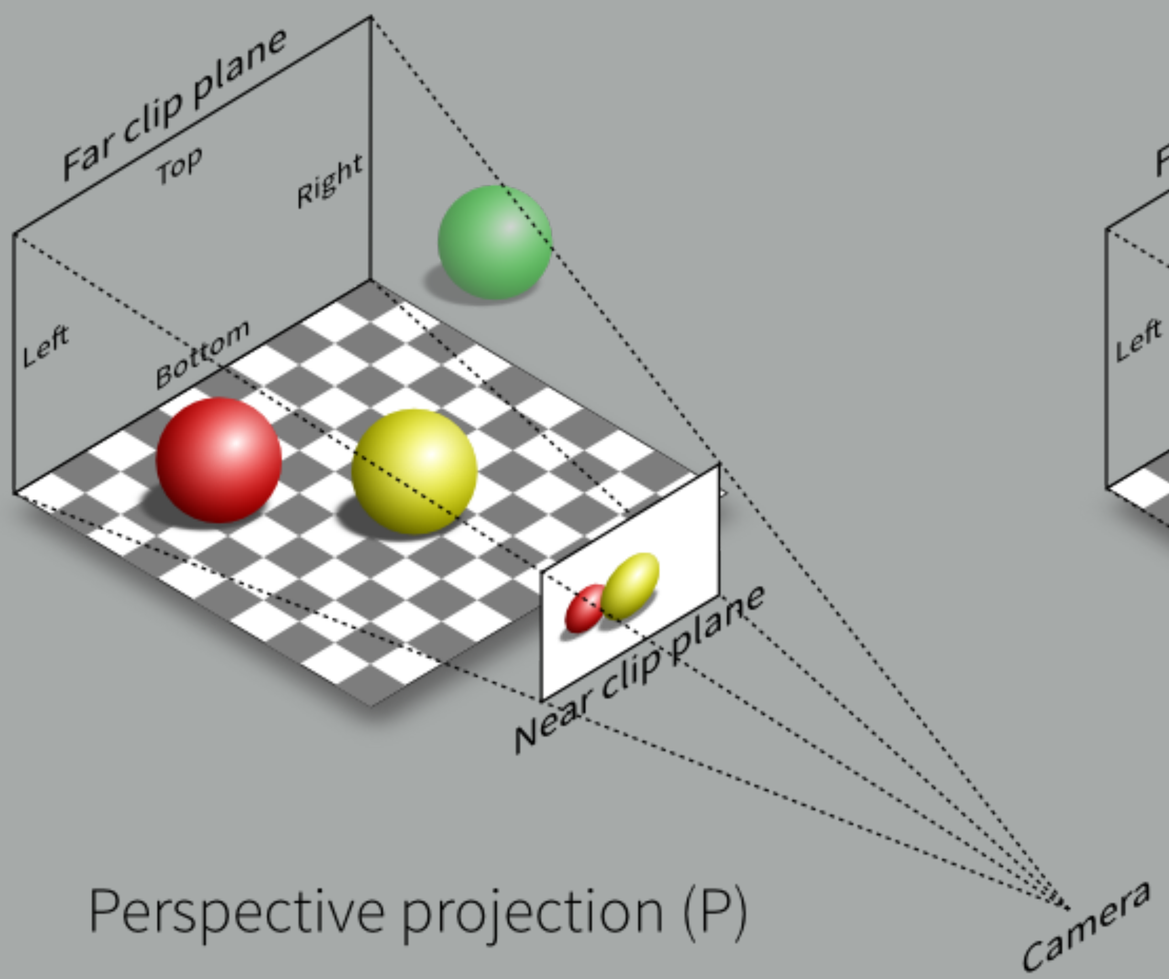
# **Projection** matrix

# What is **projection**?

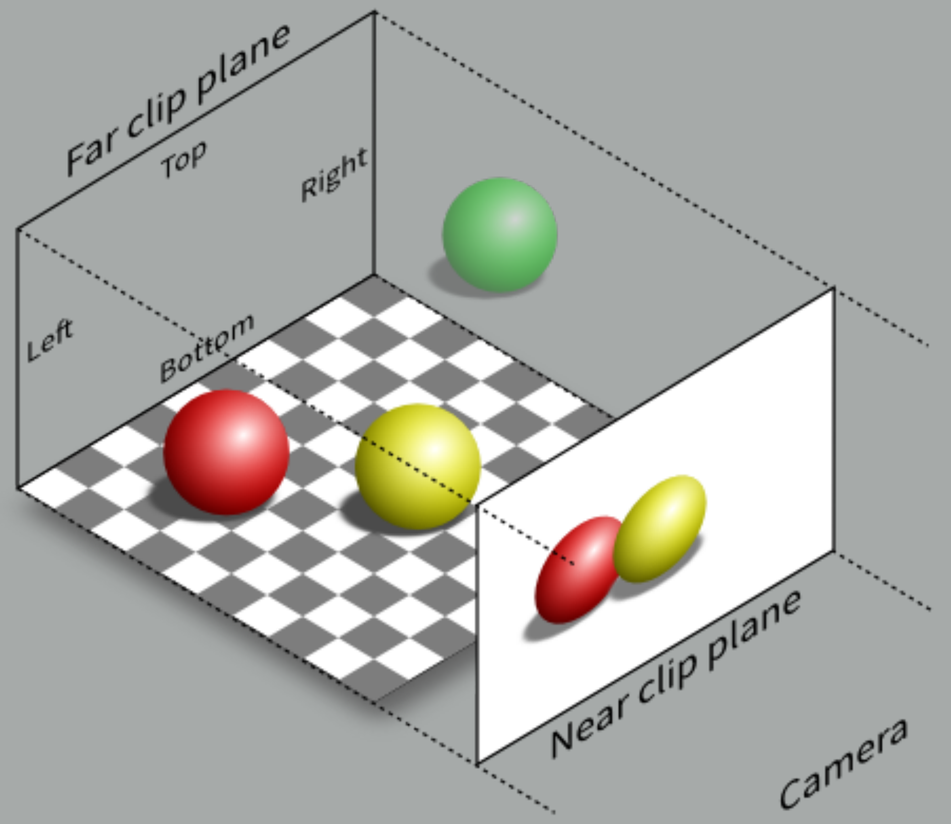
A means of representing a **three-dimensional** object in **two-dimensions**.



# **Perspective** vs. **Orthographic** projection

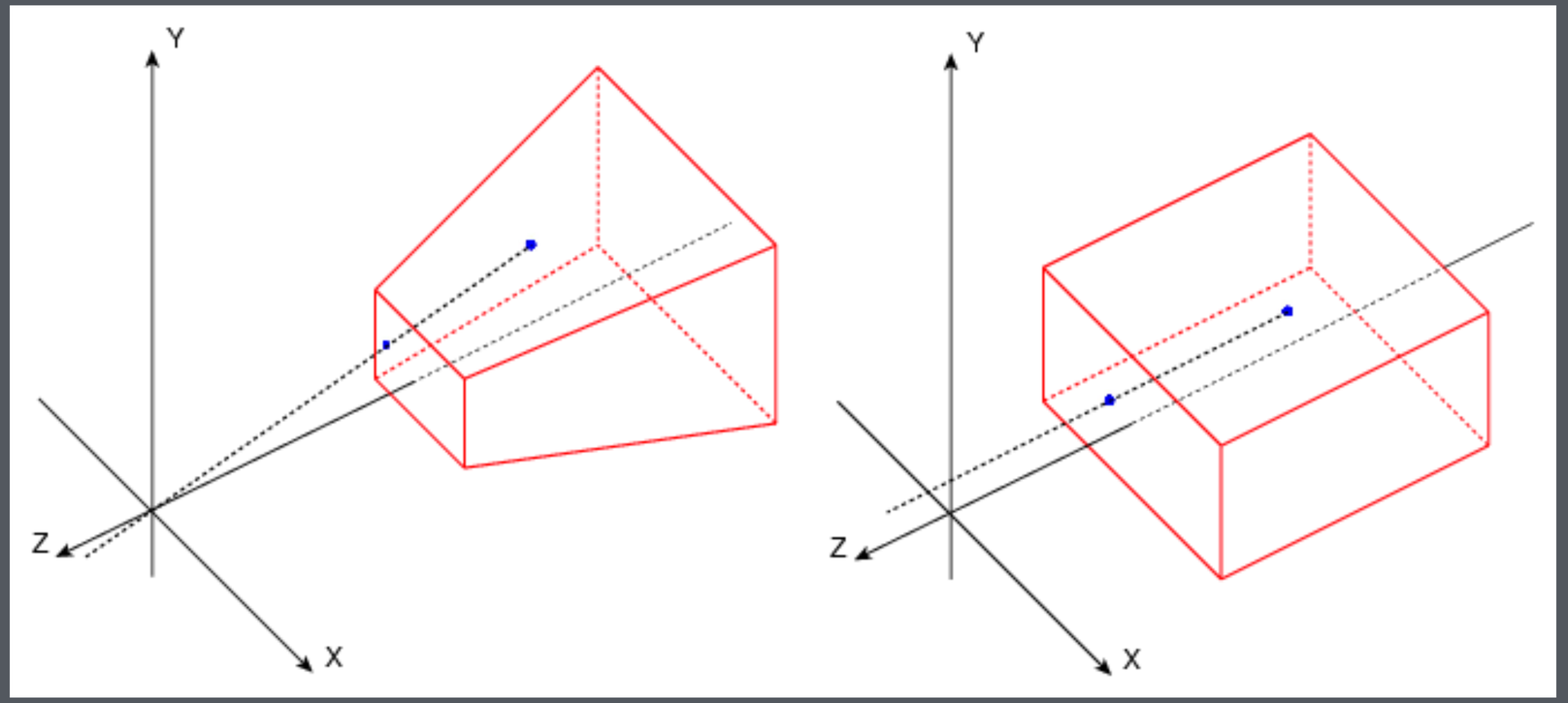


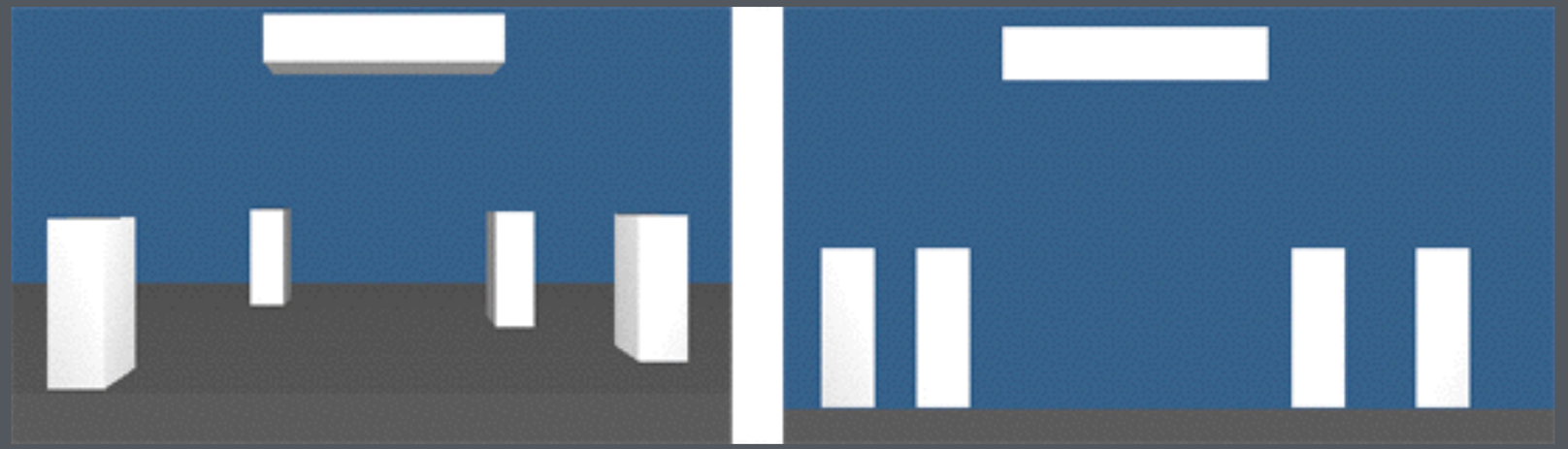

Perspective projection (P)



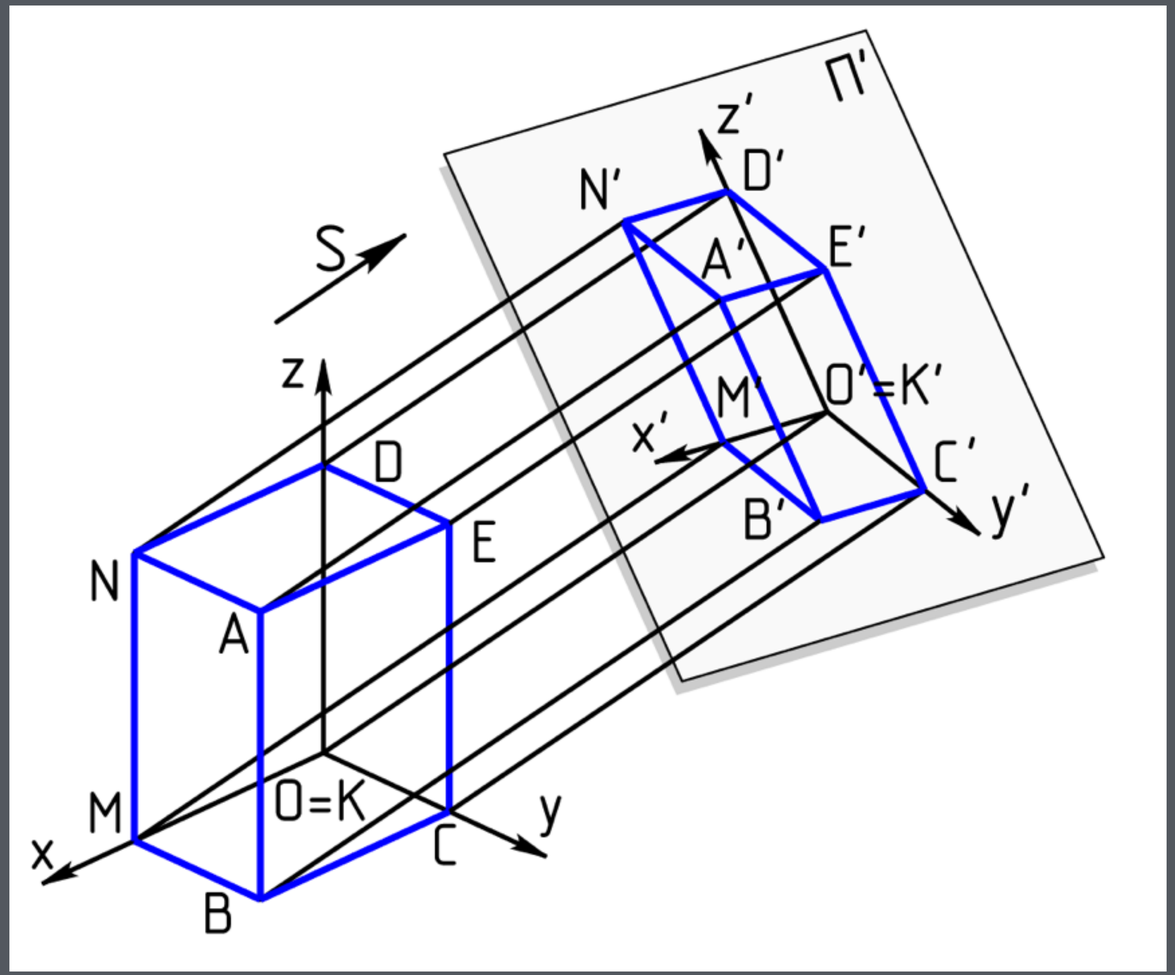

Orthographic projection (O)

# **Perspective** vs. **Orthographic** projection



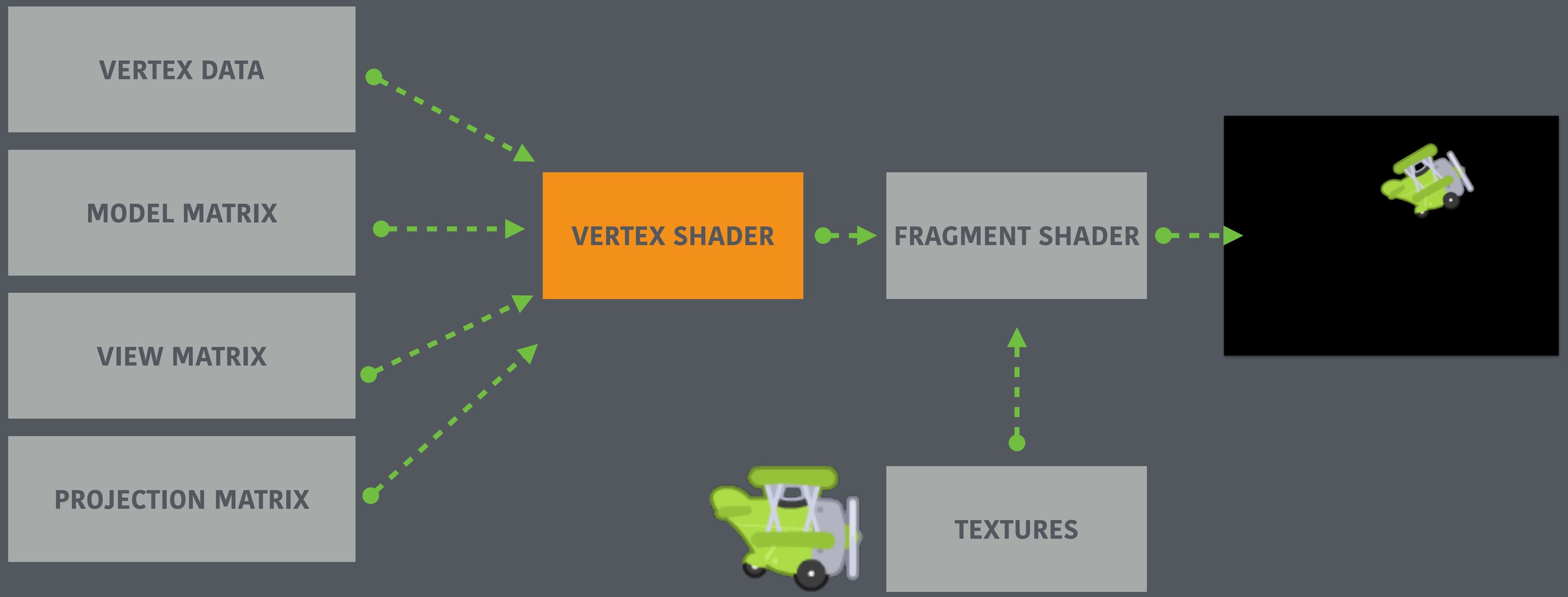



# **Projection** matrix




**Vertex position * projection matrix = screen vertex position**

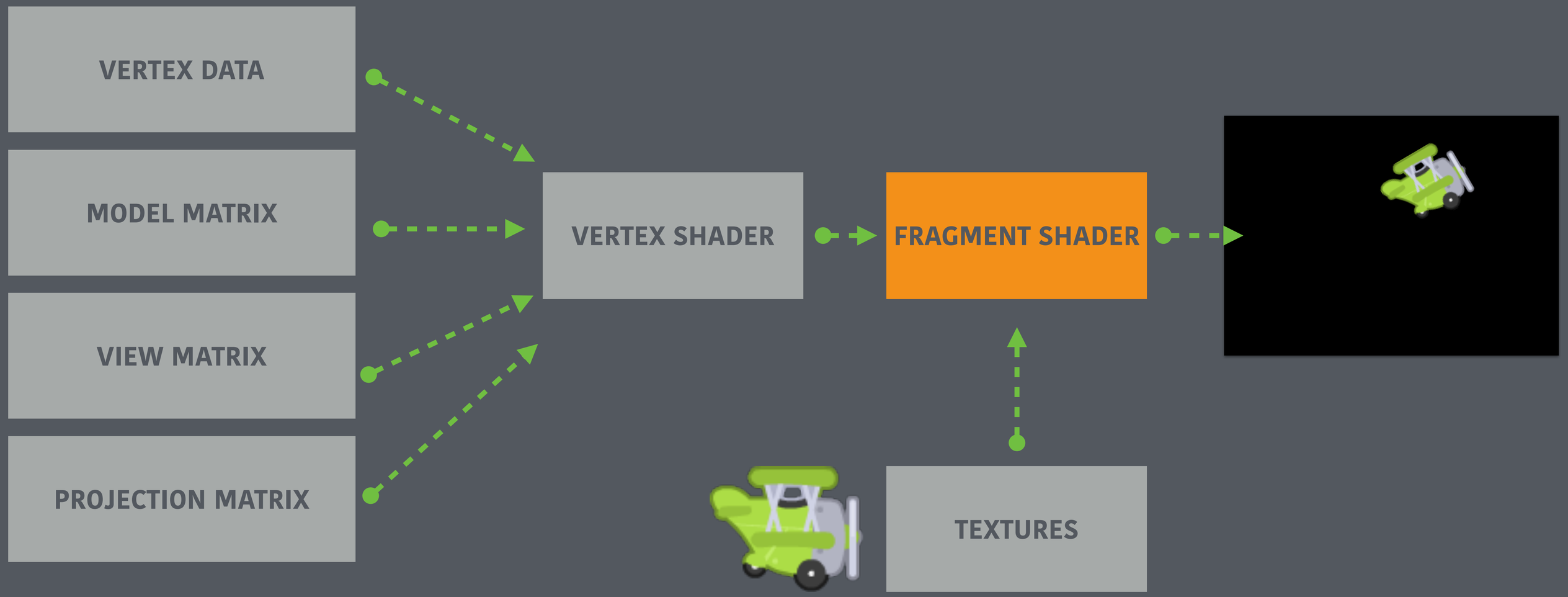
# The **GPU** pipeline



# The **vertex** shader

A **program** that transforms the **attributes**
(such as position, color or others)
of **every vertex** passed to the **GPU**.

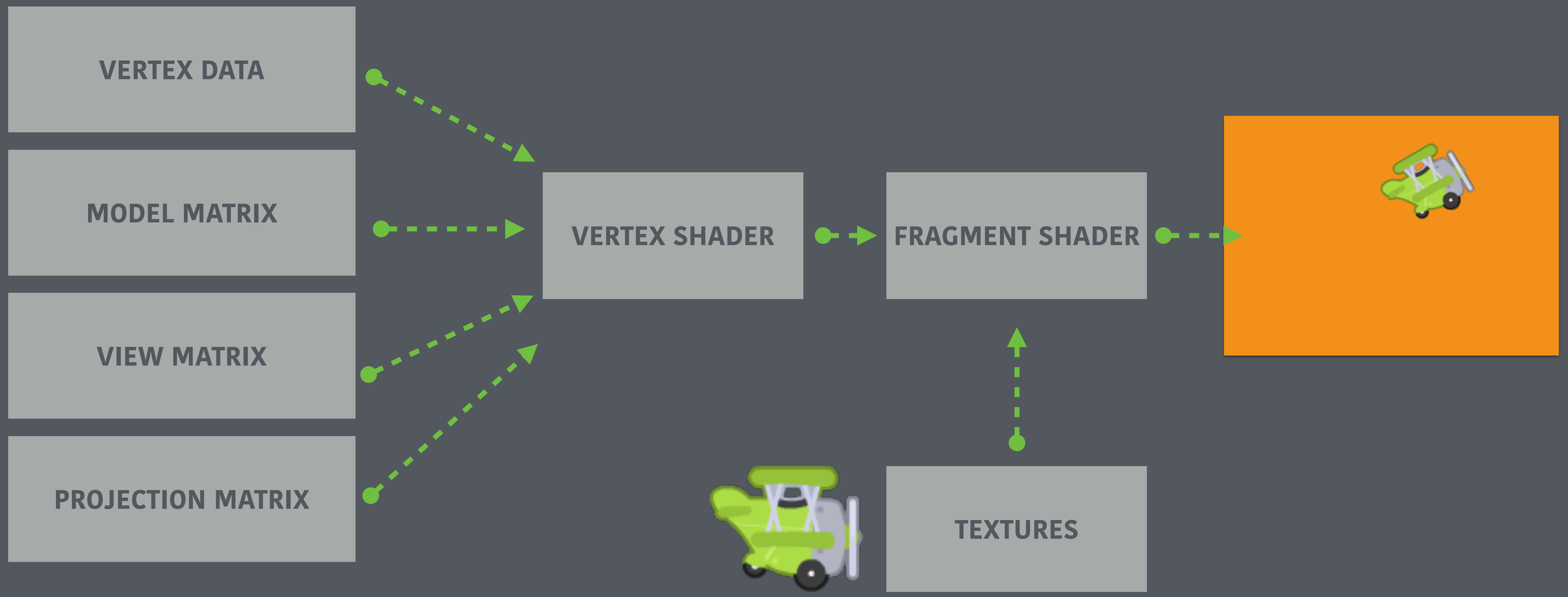
# The **GPU** pipeline

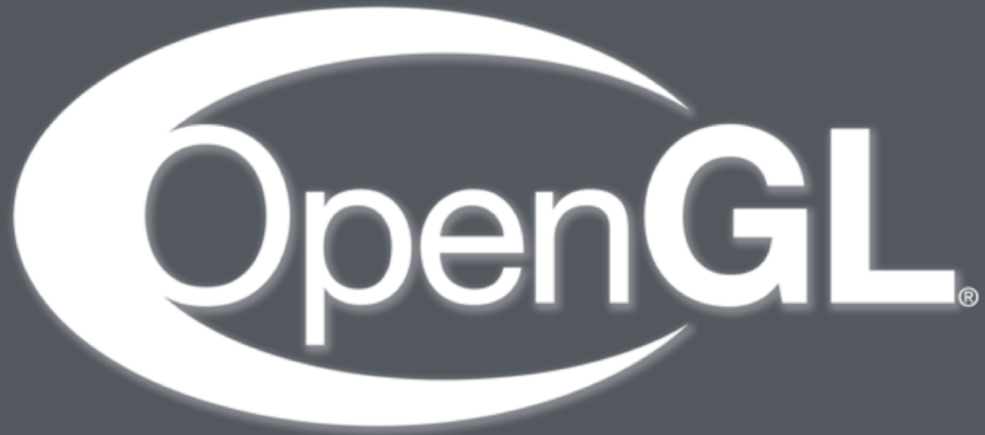


# The **fragment** shader

A **program** that returns **the color of each pixel** when geometry is rasterized on the **GPU**.

# The **GPU** pipeline




OpenGL®

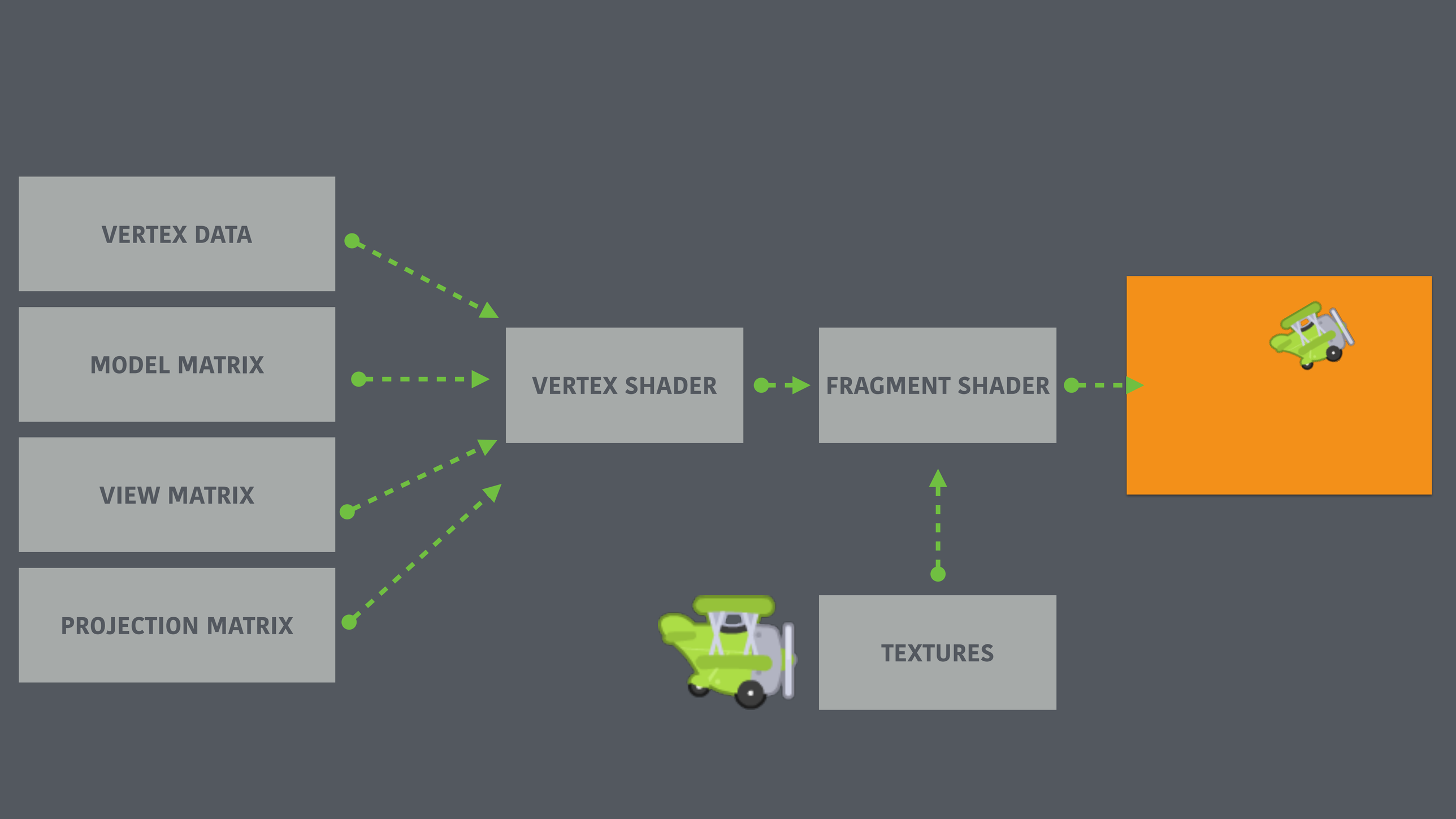
The **setup**

(happens only once!)

```
void glViewport (GLint x, GLint y, GLsizei width, GLsizei height);
```

Sets the **pixel size** and **offset** of rendering area.

```
glViewport(0, 0, 640, 360);
```


VERTEX DATA
MODEL MATRIX
VIEW MATRIX
PROJECTION MATRIX
VERTEX SHADER
FRAGMENT SHADER
TEXTURES

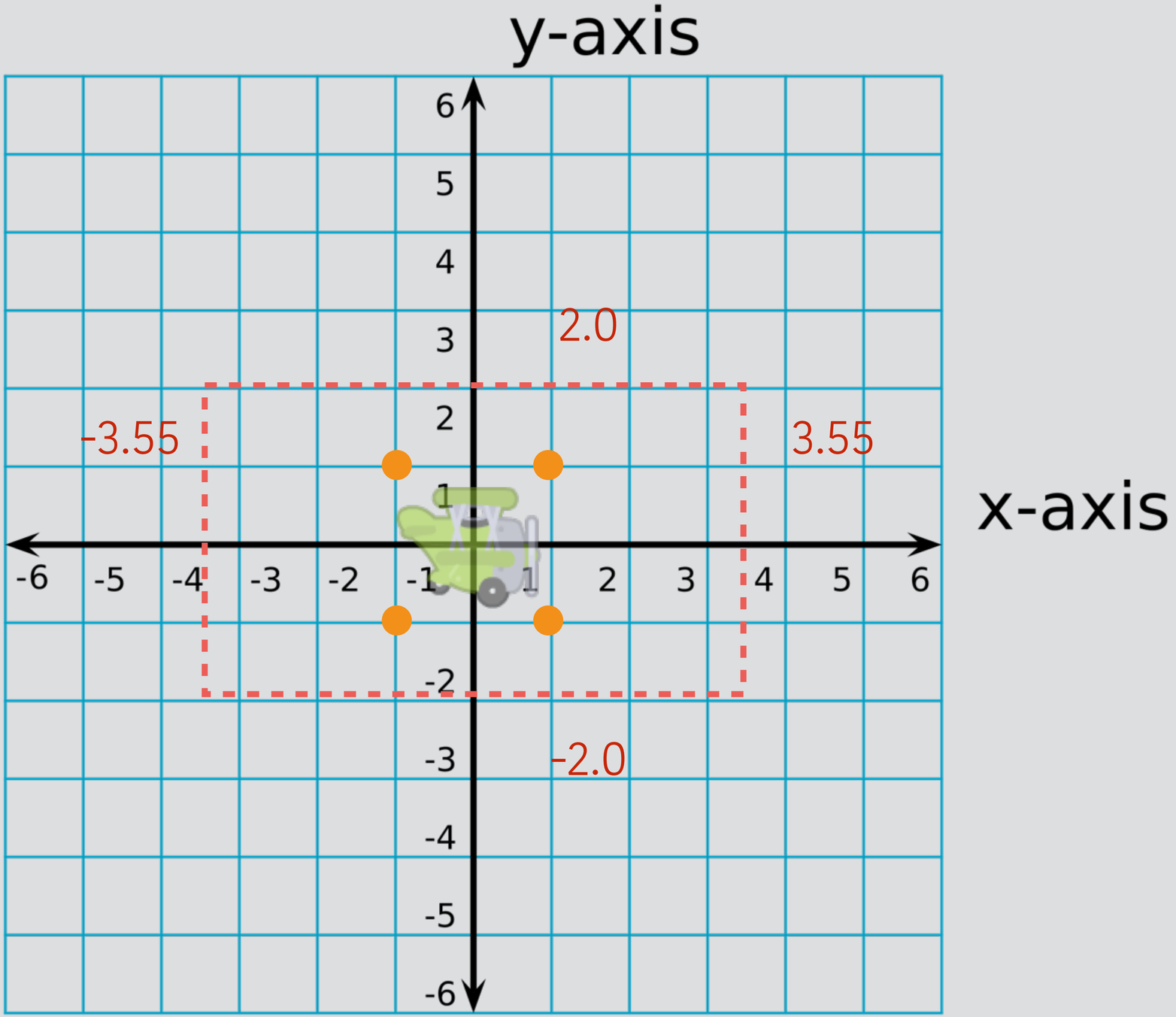
# The **Matrix** class.

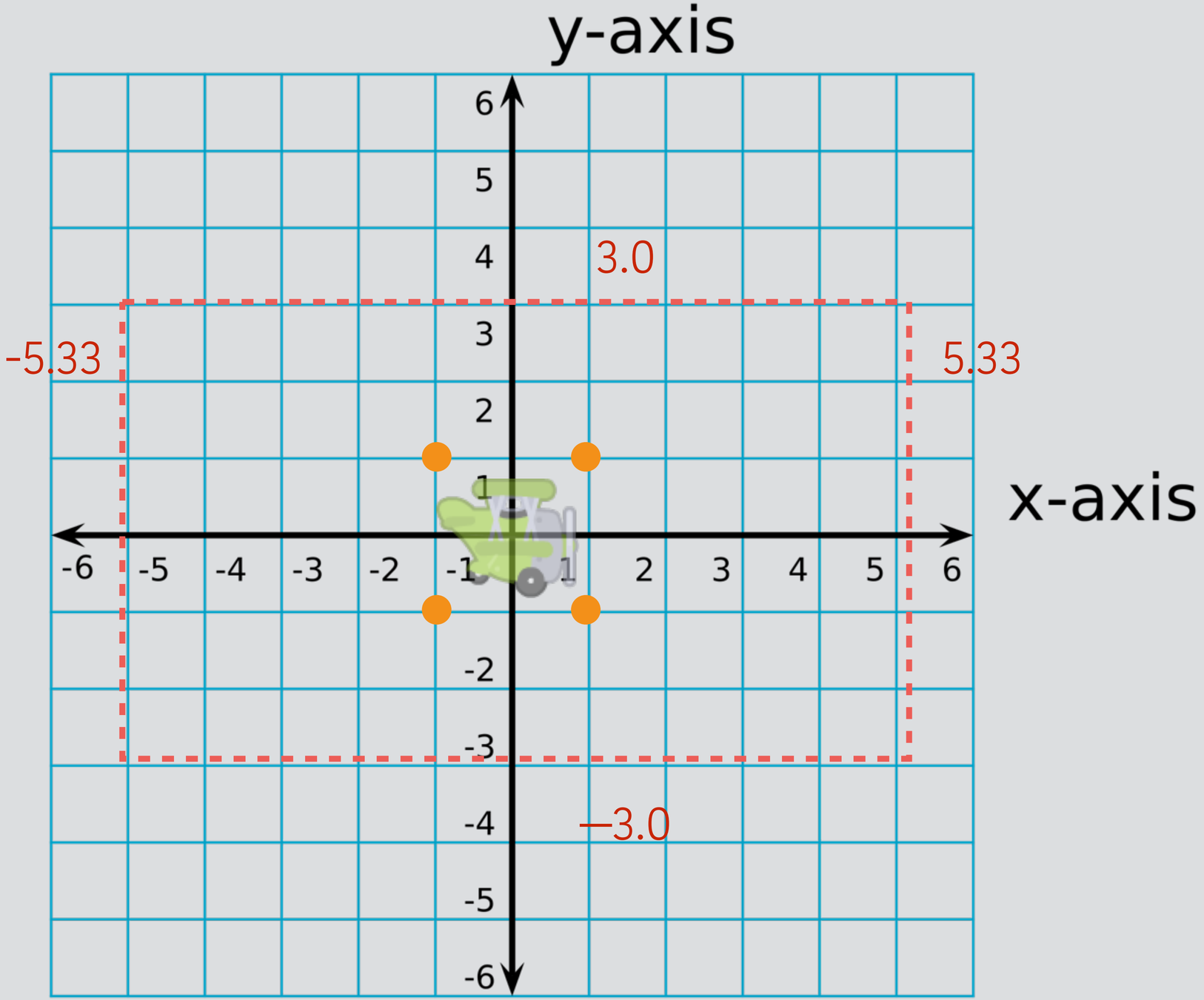
```
#include "Matrix.h"

// …

Matrix projectionMatrix;
Matrix modelMatrix;
Matrix viewMatrix;

projectionMatrix.setOrthoProjection(-1.7777f, 1.77777f, -1.0f, 1.0f, -1.0f, 1.0f);
```

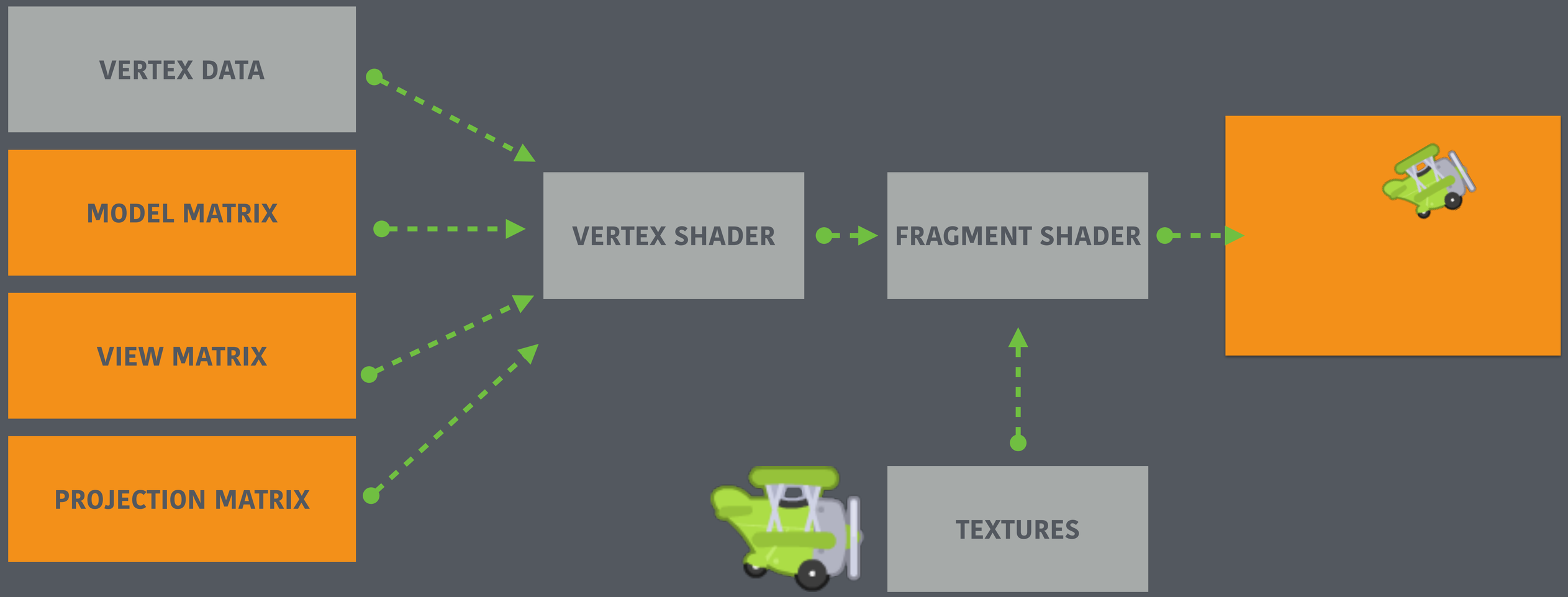
```
void Matrix::setOrthoProjection (float left, float right,
float bottom, float top, float near, float far);
```

Sets an **orthographic projection** in a **matrix.**

```
Matrix projectionMatrix;
projectionMatrix.setOrthoProjection(-3.55, 3.55, -2.0f, 2.0f, -1.0f, 1.0f);
```

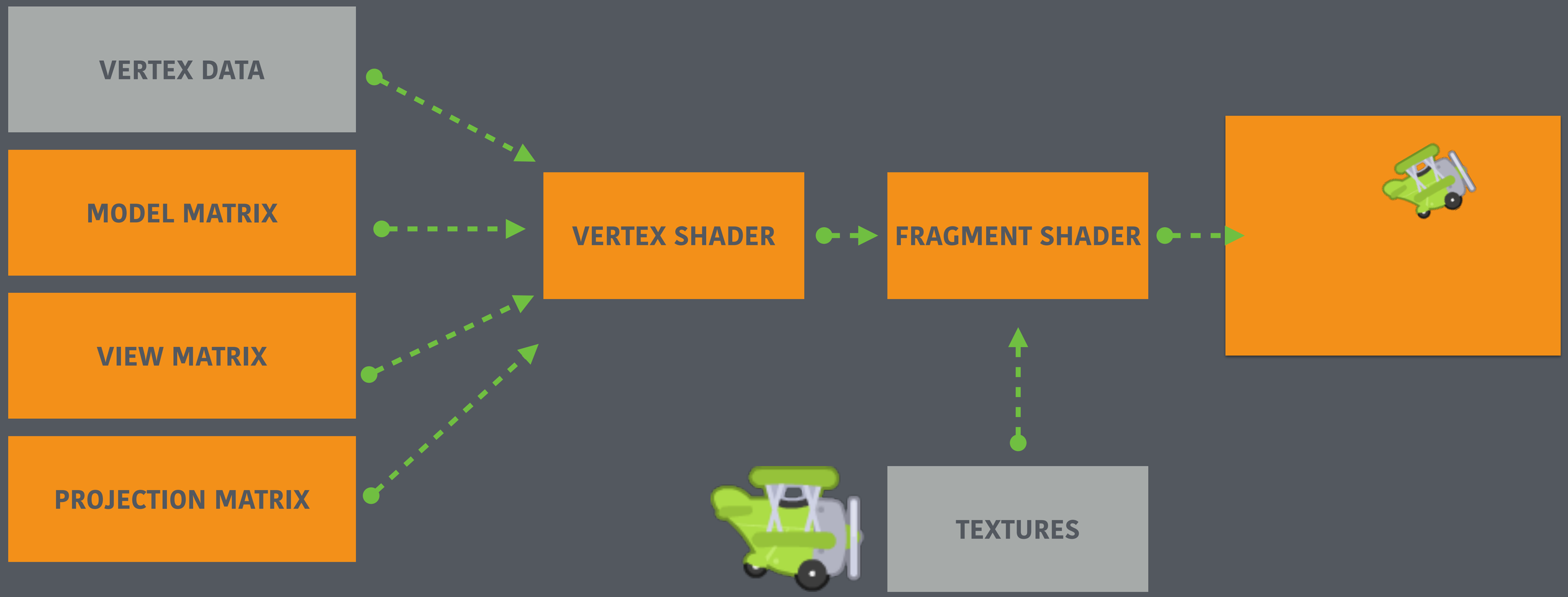

y-axis
x-axis
2.0
-3.55
3.55
-2.0
6
5
4
3
2
1
-1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6


y-axis
x-axis
3.0
-5.33
5.33
−3.0
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6


VERTEX DATA
MODEL MATRIX
VIEW MATRIX
PROJECTION MATRIX
VERTEX SHADER
FRAGMENT SHADER
TEXTURES

# The **ShaderProgram** class.

```
#include "ShaderProgram.h"

// …

ShaderProgram program(RESOURCE_FOLDER"vertex.glsl", RESOURCE_FOLDER"fragment.glsl");
```


VERTEX DATA
MODEL MATRIX
VIEW MATRIX
PROJECTION MATRIX
VERTEX SHADER
FRAGMENT SHADER
TEXTURES

Drawing **polygons**.
(happens every frame!)

# Pass the **matrices** to our **program**.

```
program.setModelMatrix(modelMatrix);
program.setProjectionMatrix(projectionMatrix);
program.setViewMatrix(viewMatrix);
```

```
void glUseProgram (GLint programID);
```

Use the specified **program id**.

```
glUseProgram(program.programID);
```

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

Defines an array of **vertex data**.

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

glVertexAttribPointer(program.positionAttribute, 2, GL_FLOAT, false, 0, vertices);
```

```
void glEnableVertexAttribArray (GLuint index);
```

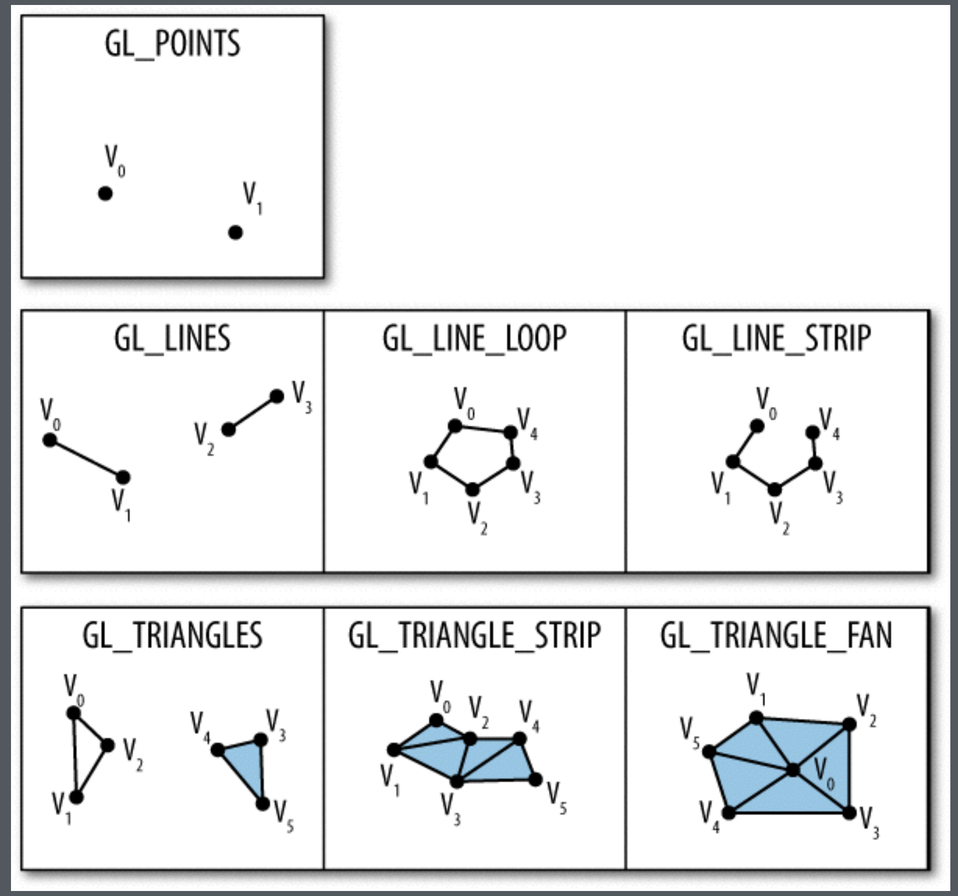
Enables a **vertex attribute array**.

```
glEnableVertexAttribArray(program.positionAttribute);
```

```
glDrawArrays (GLenum mode, GLint first,
GLsizei count);
```

Render previously defined **arrays**.

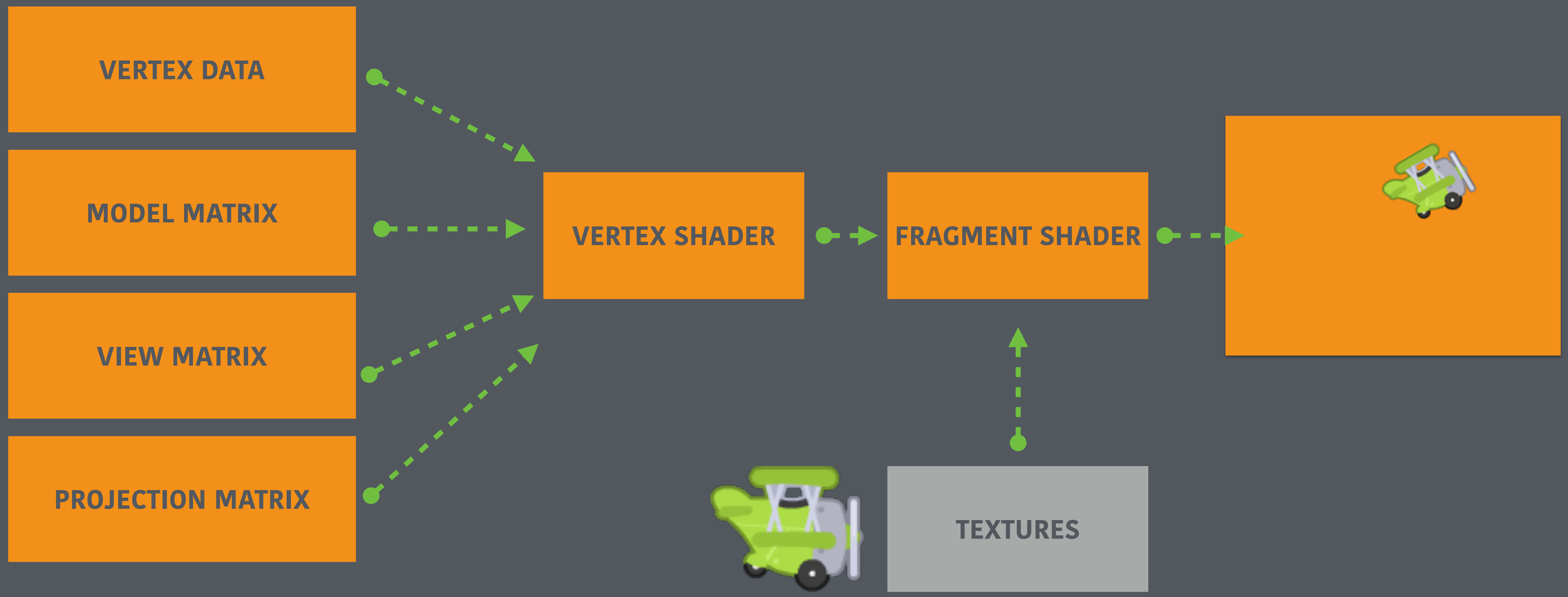
```
glDrawArrays(GL_TRIANGLES, 0, 3);
```



```
void glDisableVertexAttribArray (GLuint index);
```

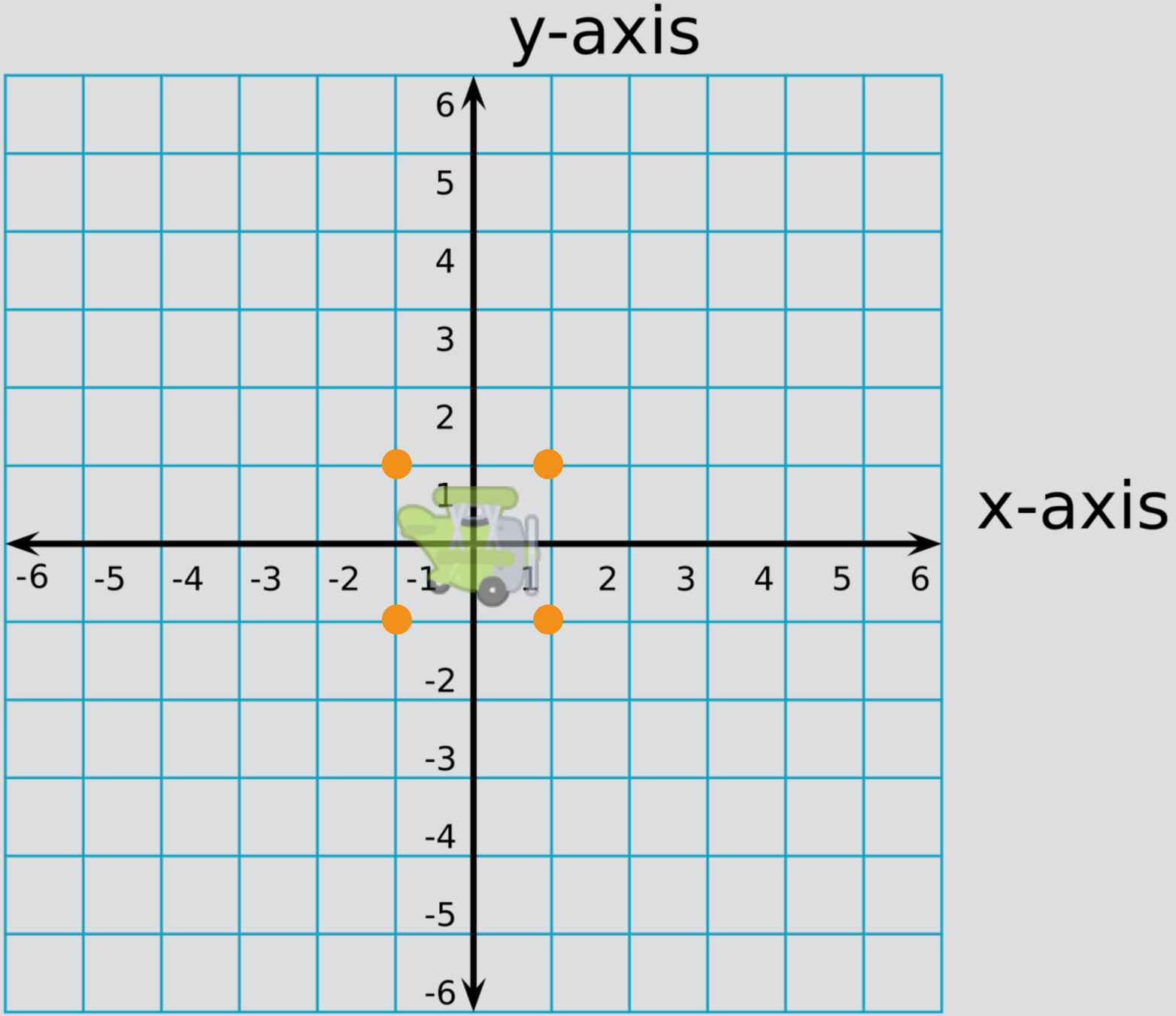
Disables a **vertex attribute array**.

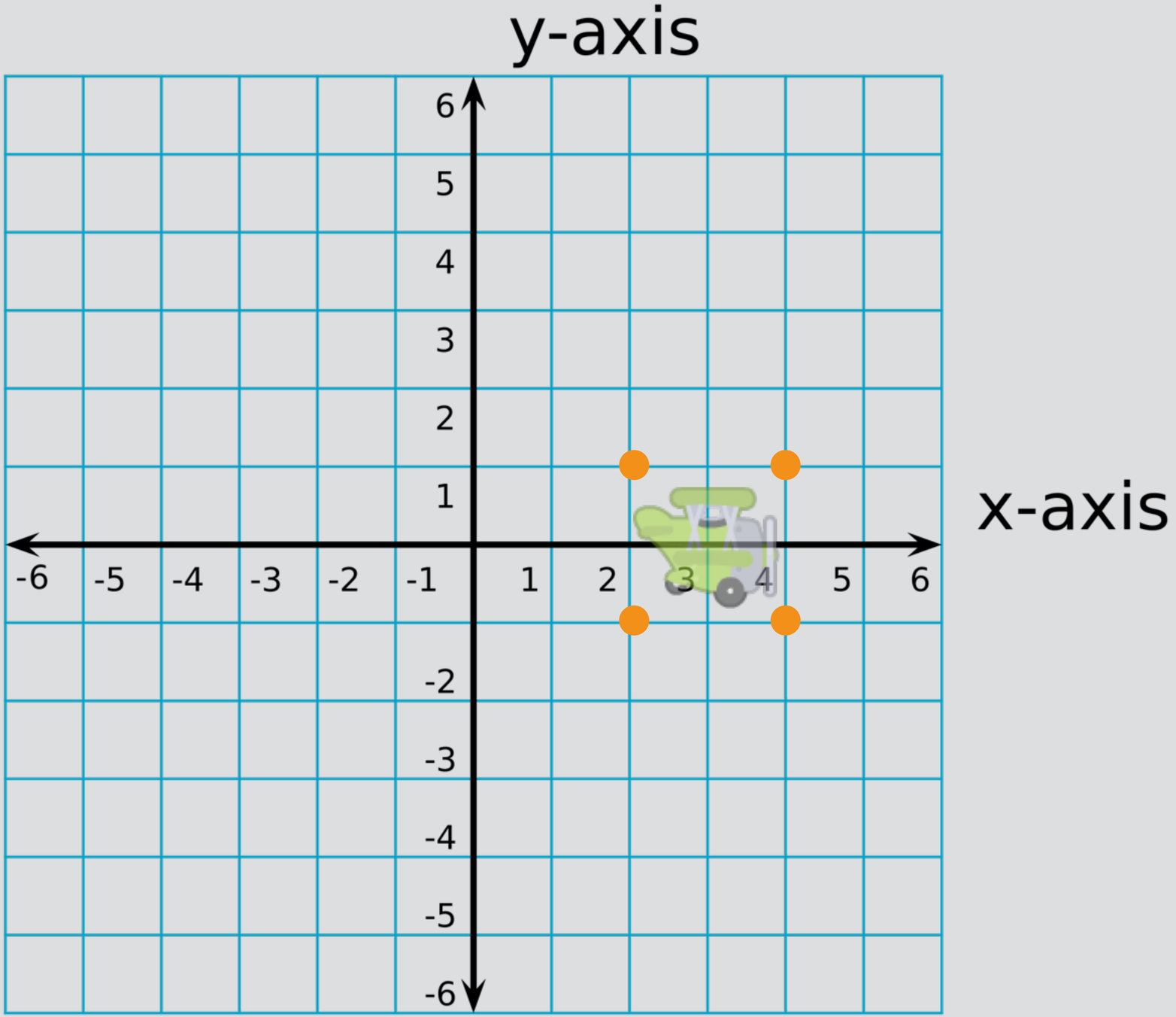
```
glEnableVertexAttribArray(program.positionAttribute);
```


VERTEX DATA
MODEL MATRIX
VIEW MATRIX
PROJECTION MATRIX
VERTEX SHADER
FRAGMENT SHADER
TEXTURES

# Transformations

# Translations


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

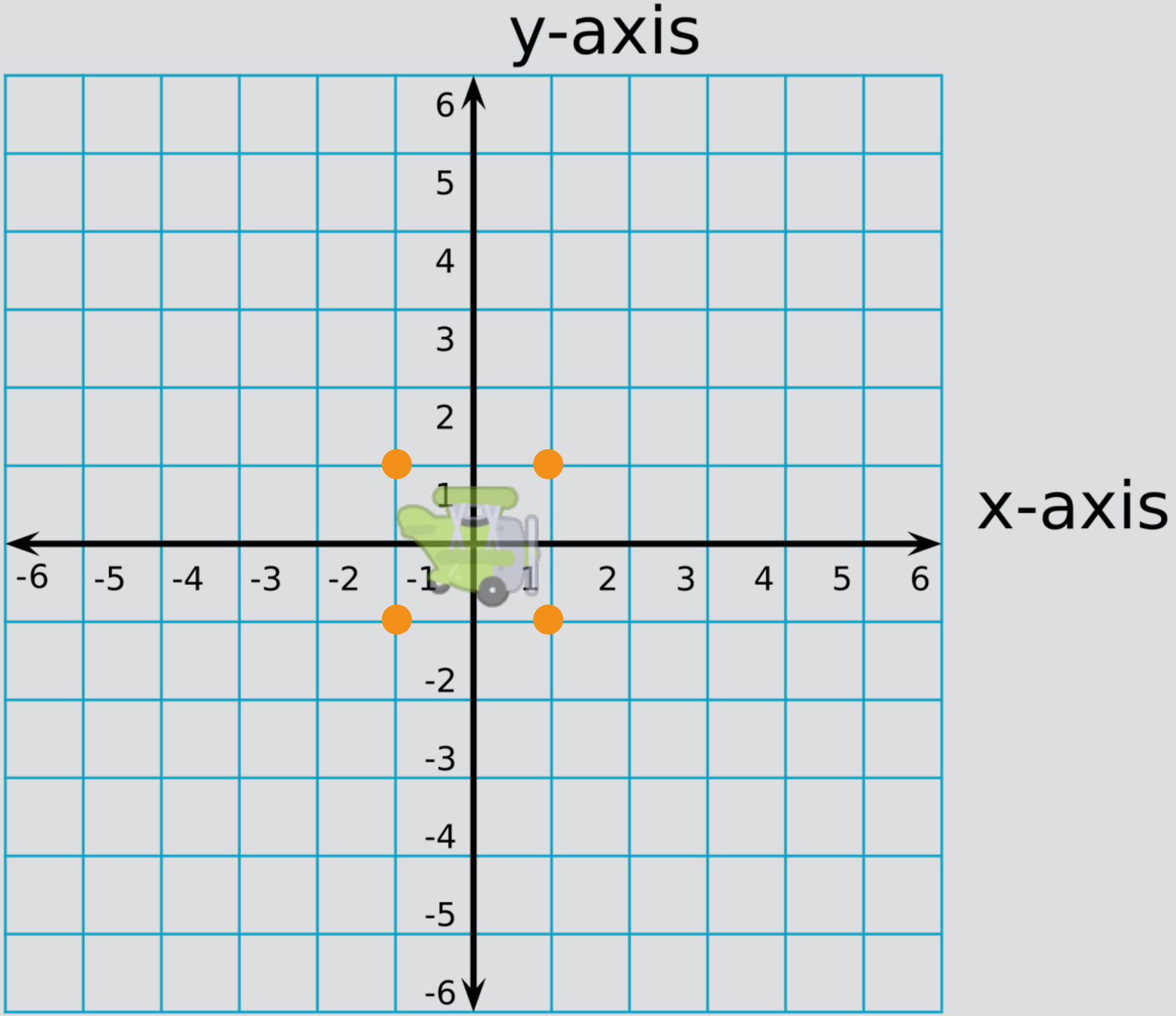

y-axis
x-axis
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6

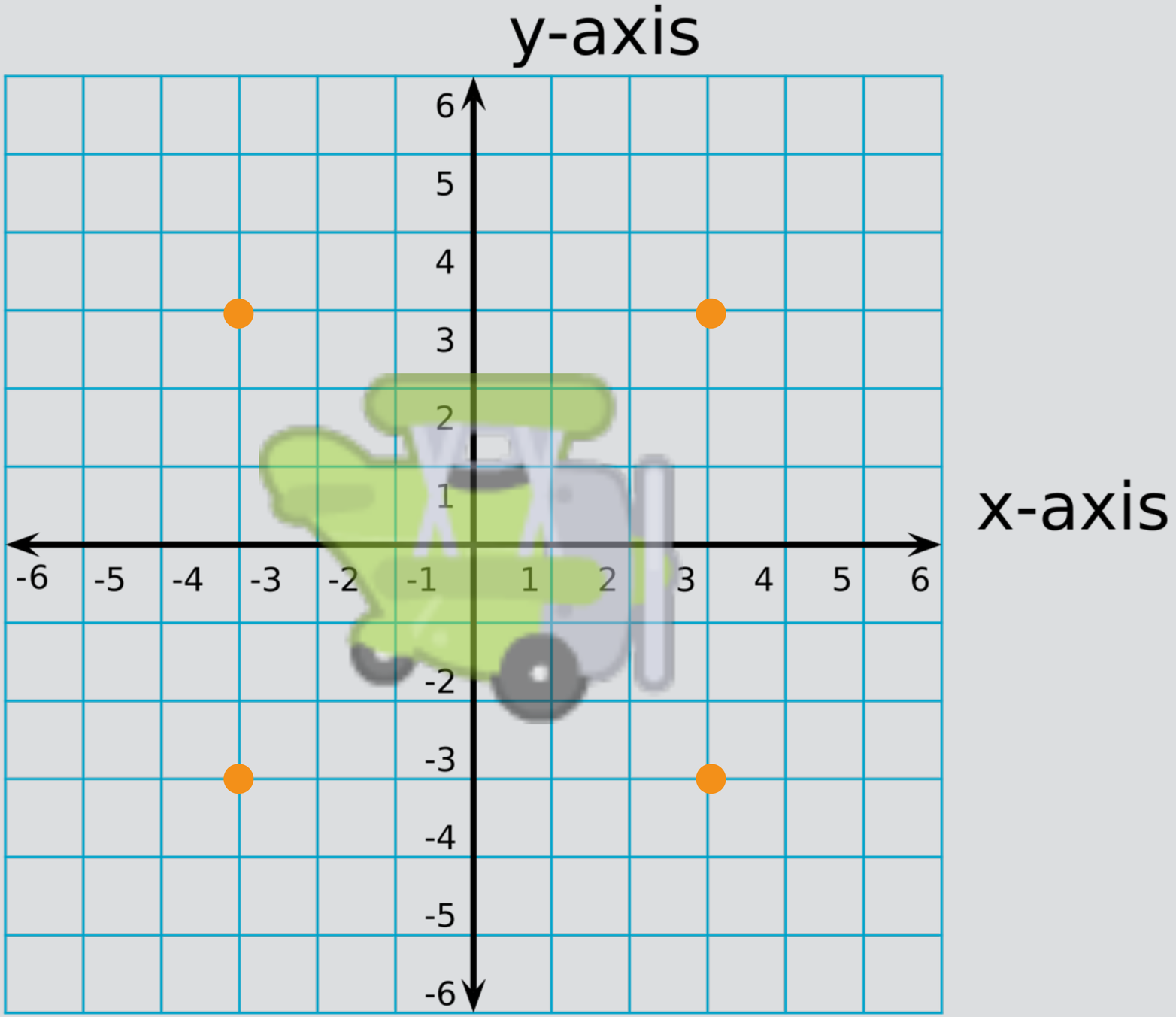
```
void Matrix::Translate (float x, float y, float z);
```

**Translates** a matrix by specified coordinates.

```
Matrix modelMatrix;
modelMatrix.Translate(1.0, 0.0, 0.0);
```

# Scaling


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

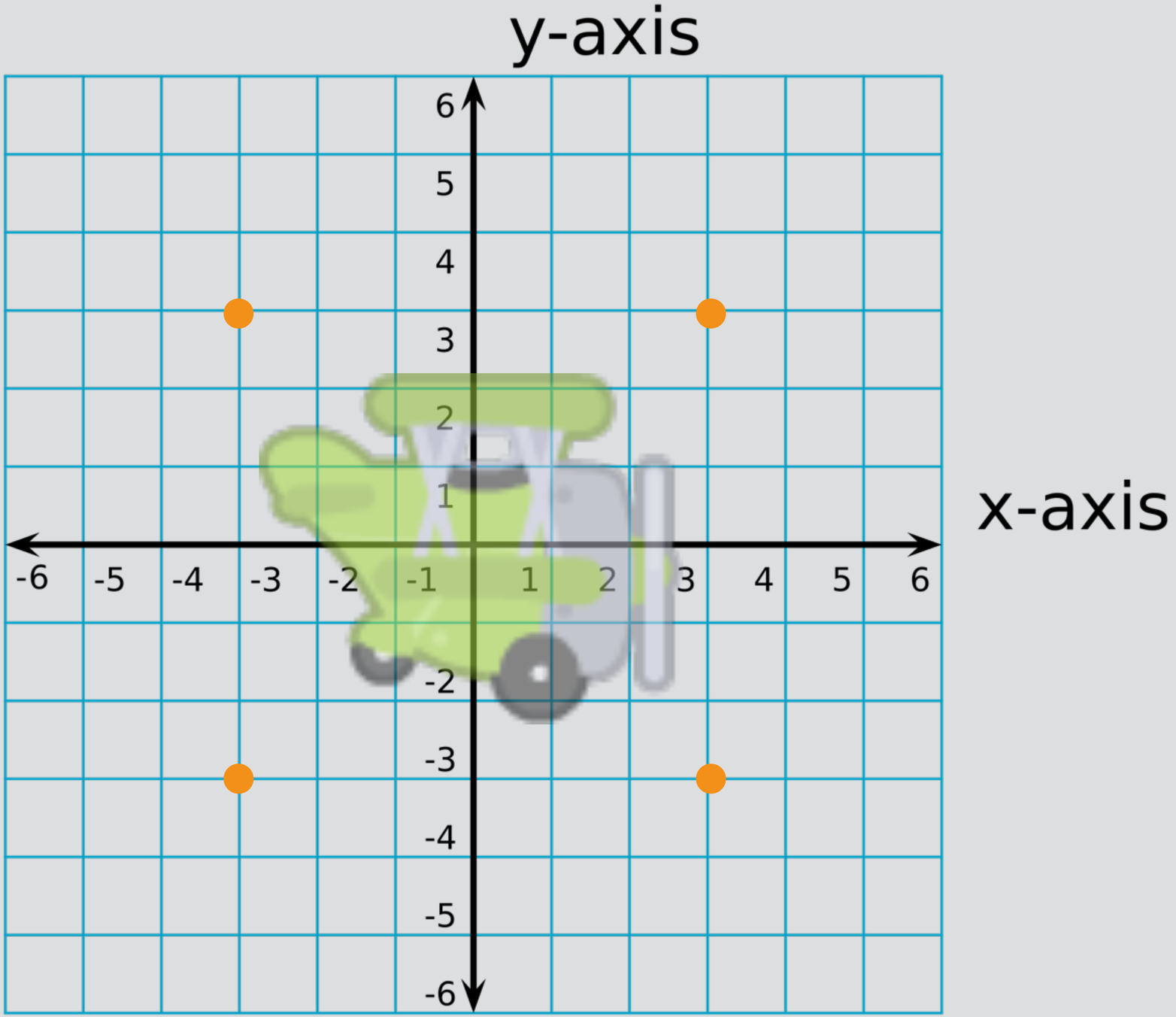

y-axis
x-axis
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6

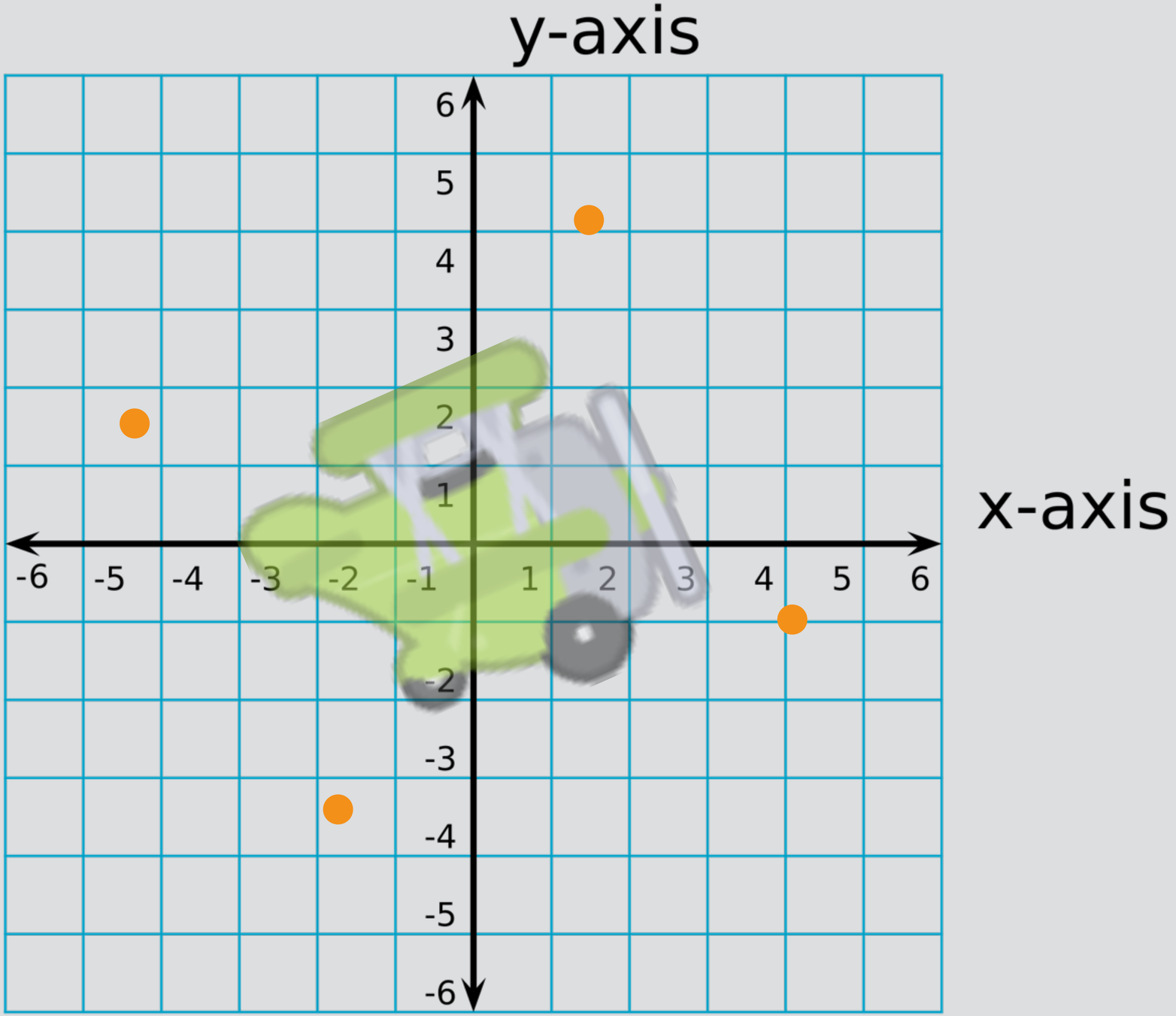
```
void Matrix::Scale (float x, float y, float z);
```

**Scales** a matrix by specified coordinates.

```
Matrix modelMatrix;
modelMatrix.Scale(2.0, 2.0, 1.0);
```

# Rotation


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


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

```
void Matrix::Rotate(float radians);
```

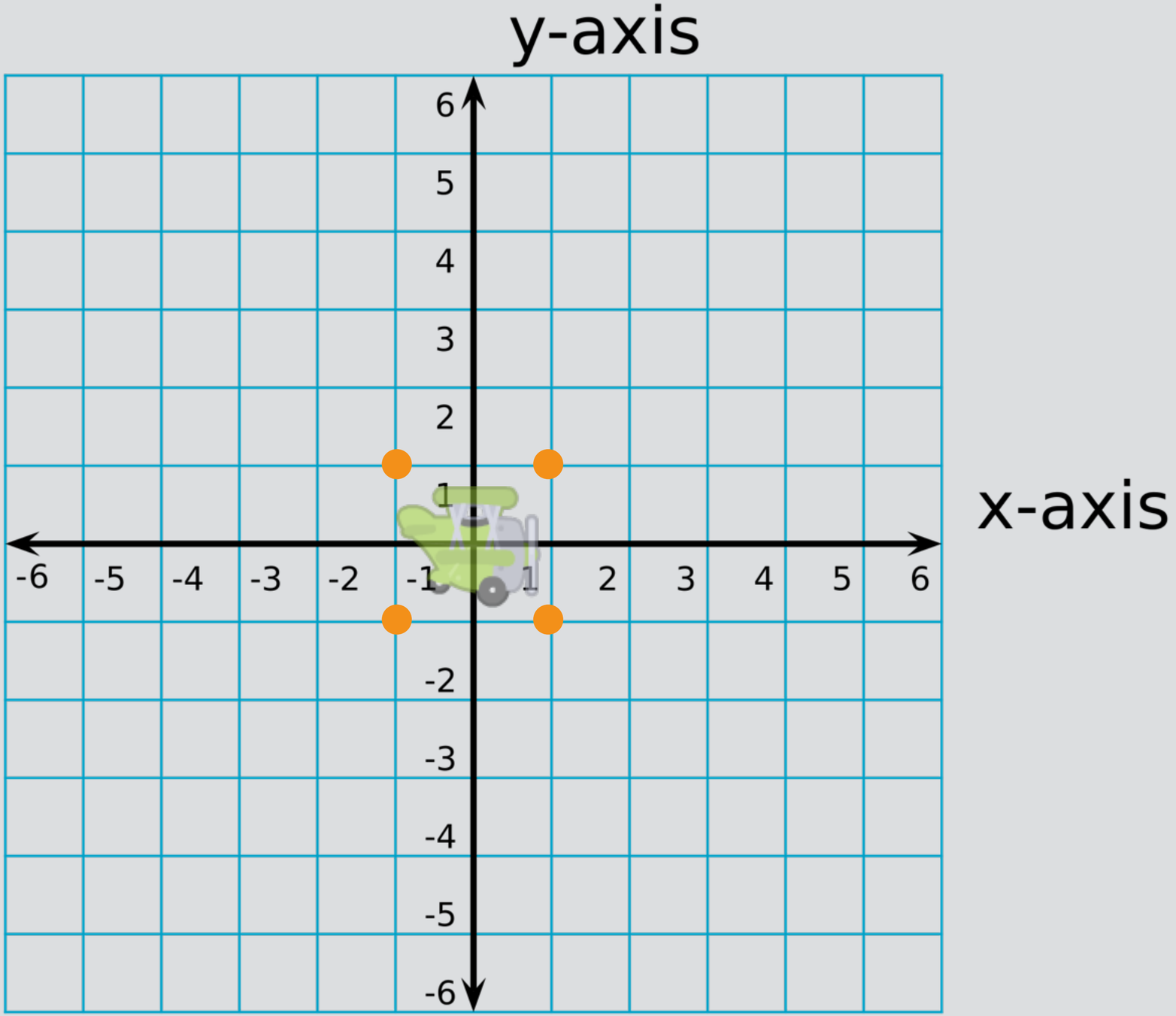
**Rotates** a matrix by specified radians.

```
Matrix modelMatrix;
modelMatrix.Rotate(45.0 * (3.1415926 / 180.0));
```

To convert from **degrees** to **radians**:

RADIANS = DEGREES * (PI / 180)

# Resetting a matrix.


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

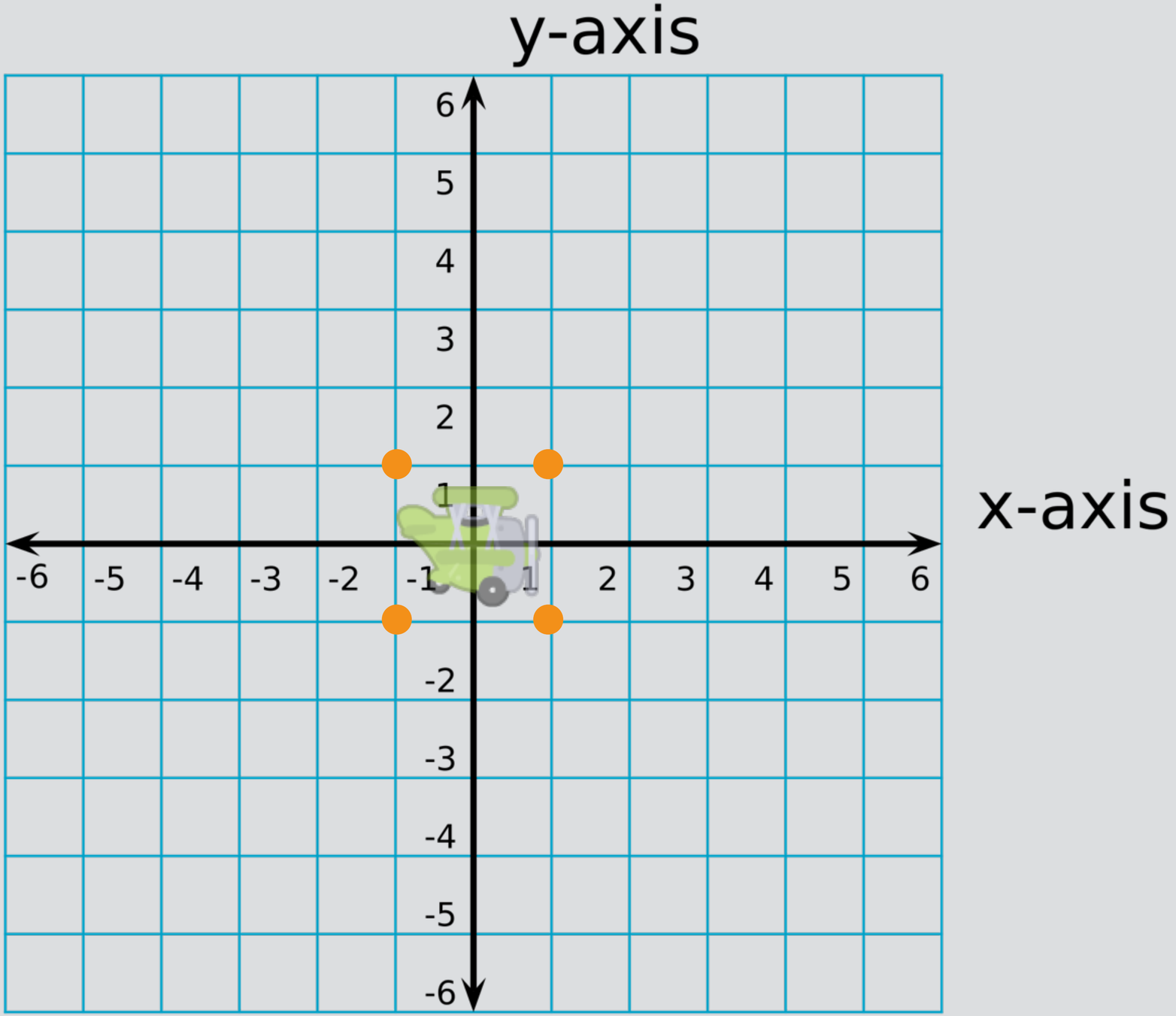
```
void Matrix::identity();
```

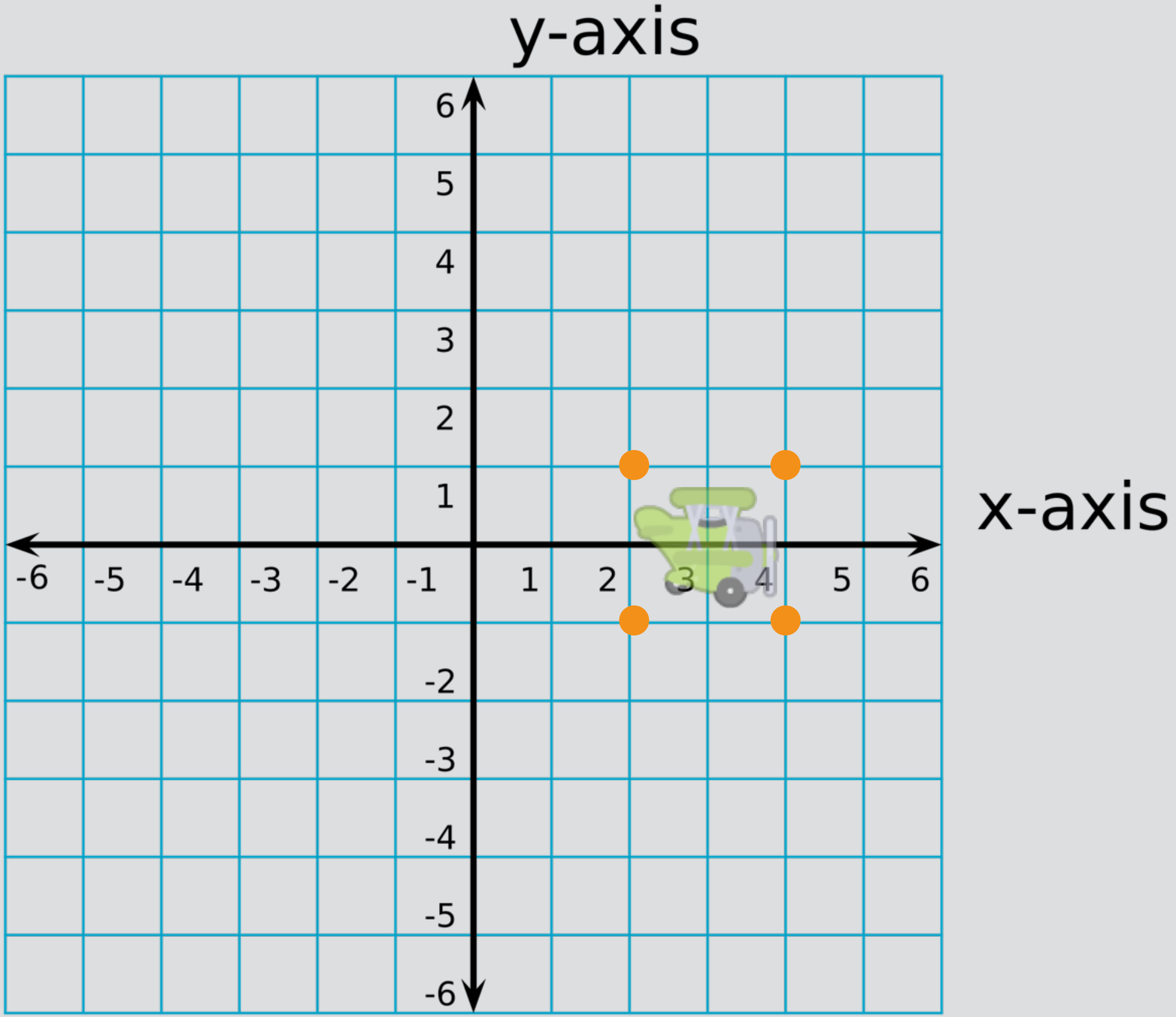
**Resets** a matrix to have no transformations.

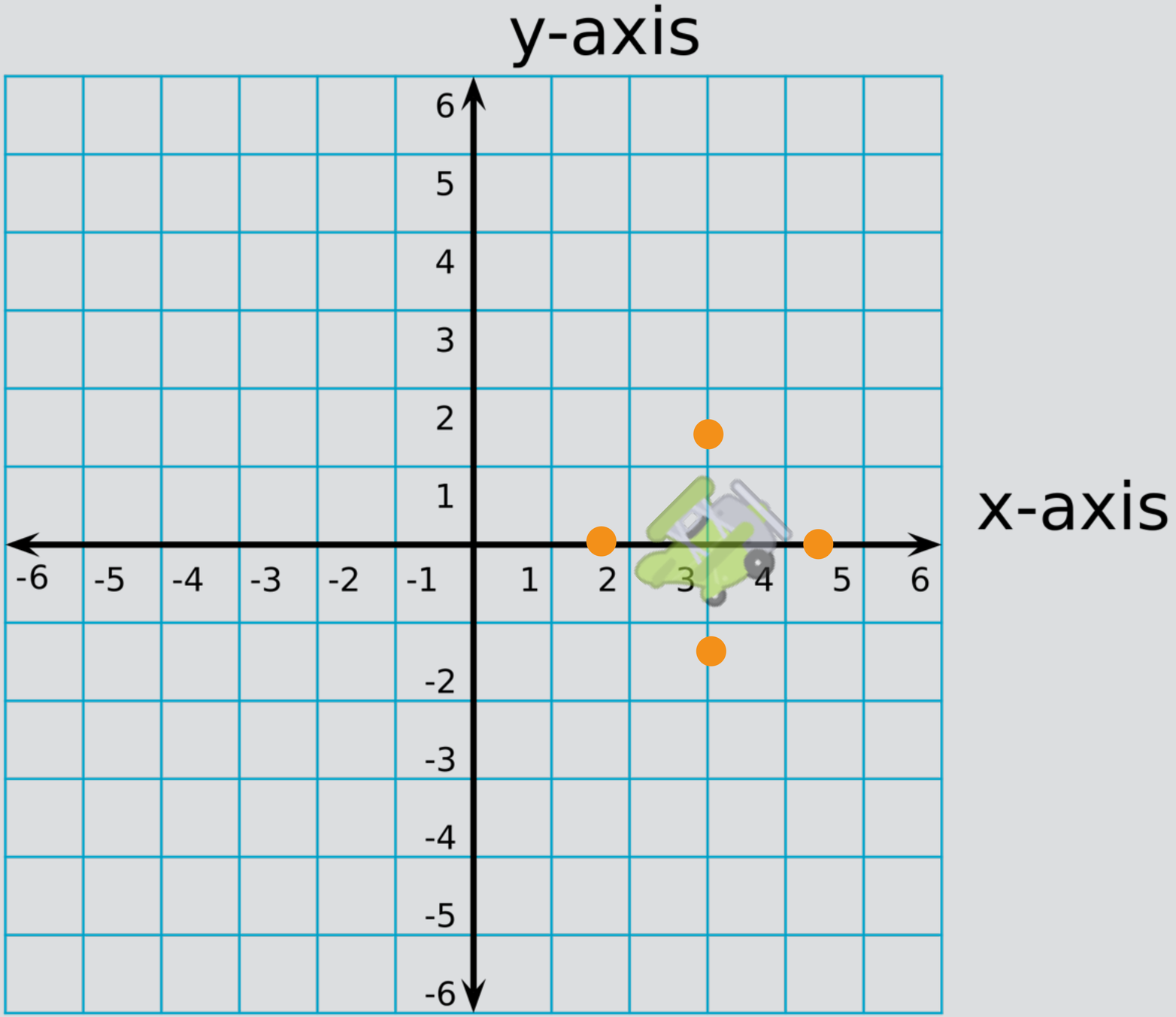
```
Matrix modelMatrix;
modelMatrix.identity();
```

Matrix transformations
are **additive!**

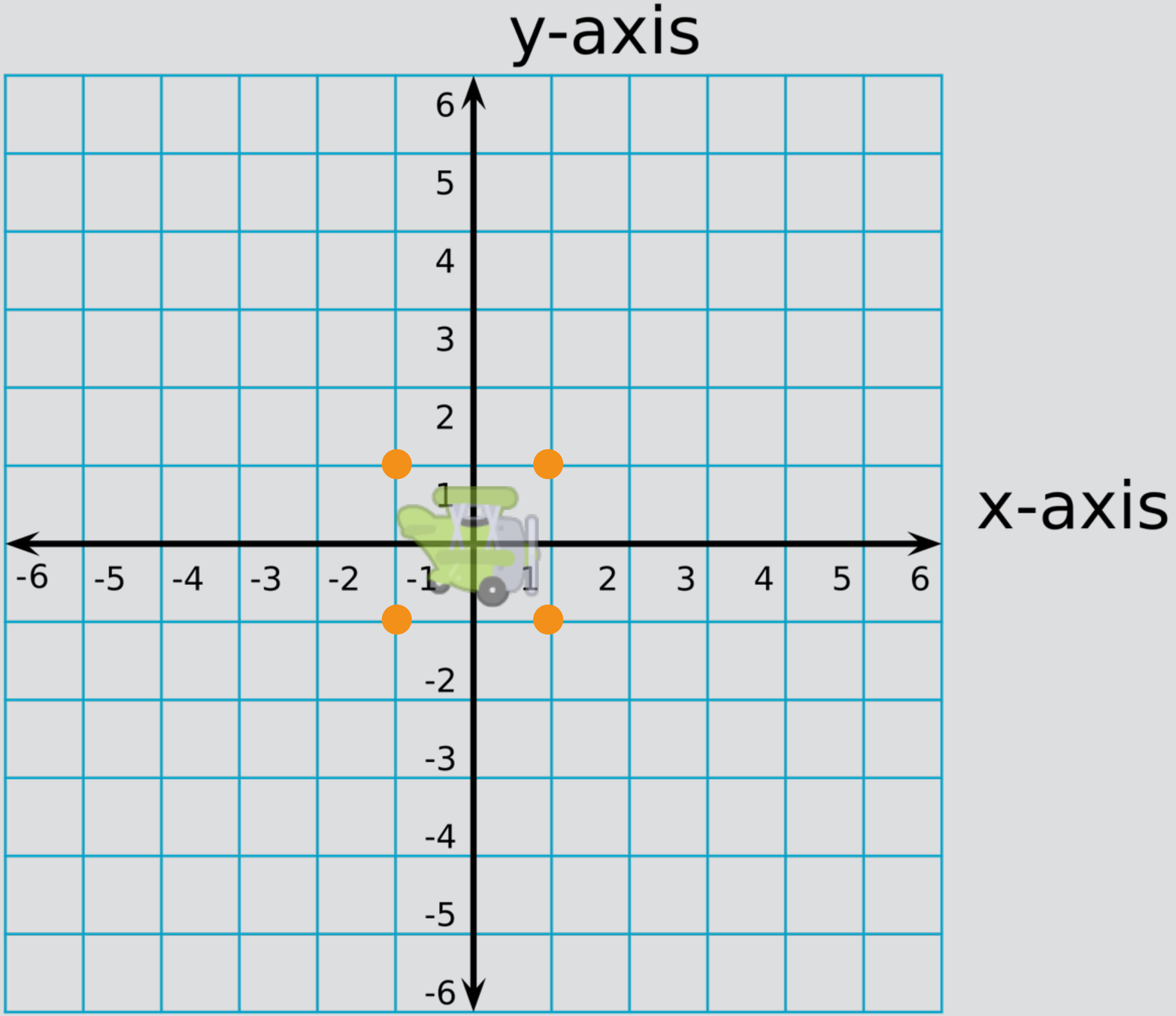
```
Matrix modelMatrix;
modelMatrix.Translate(2.0, 0.0, 0.0);
modelMatrix.Rotate(45.0 * (3.1415926 / 180.0));
```

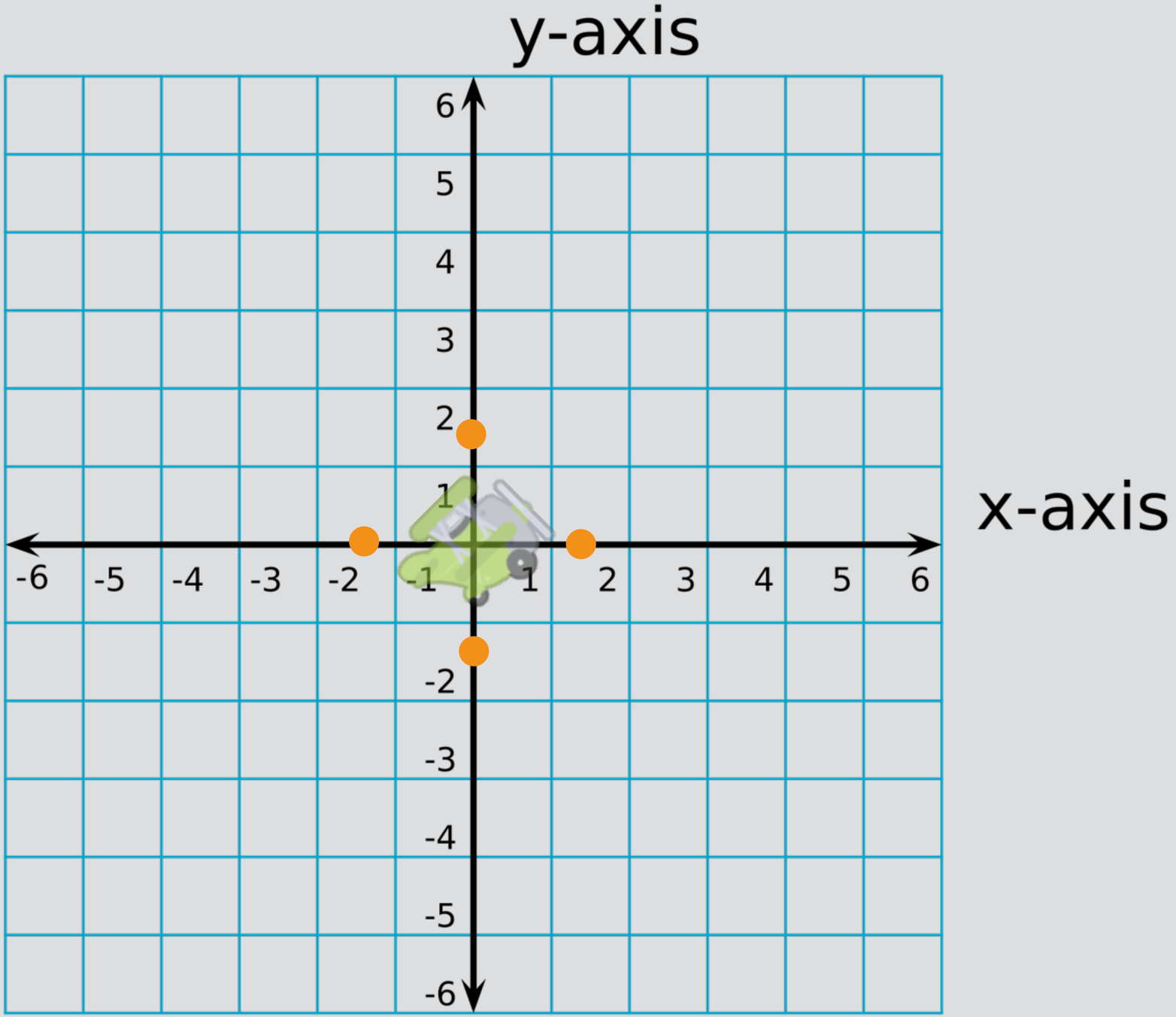

y-axis
x-axis
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6

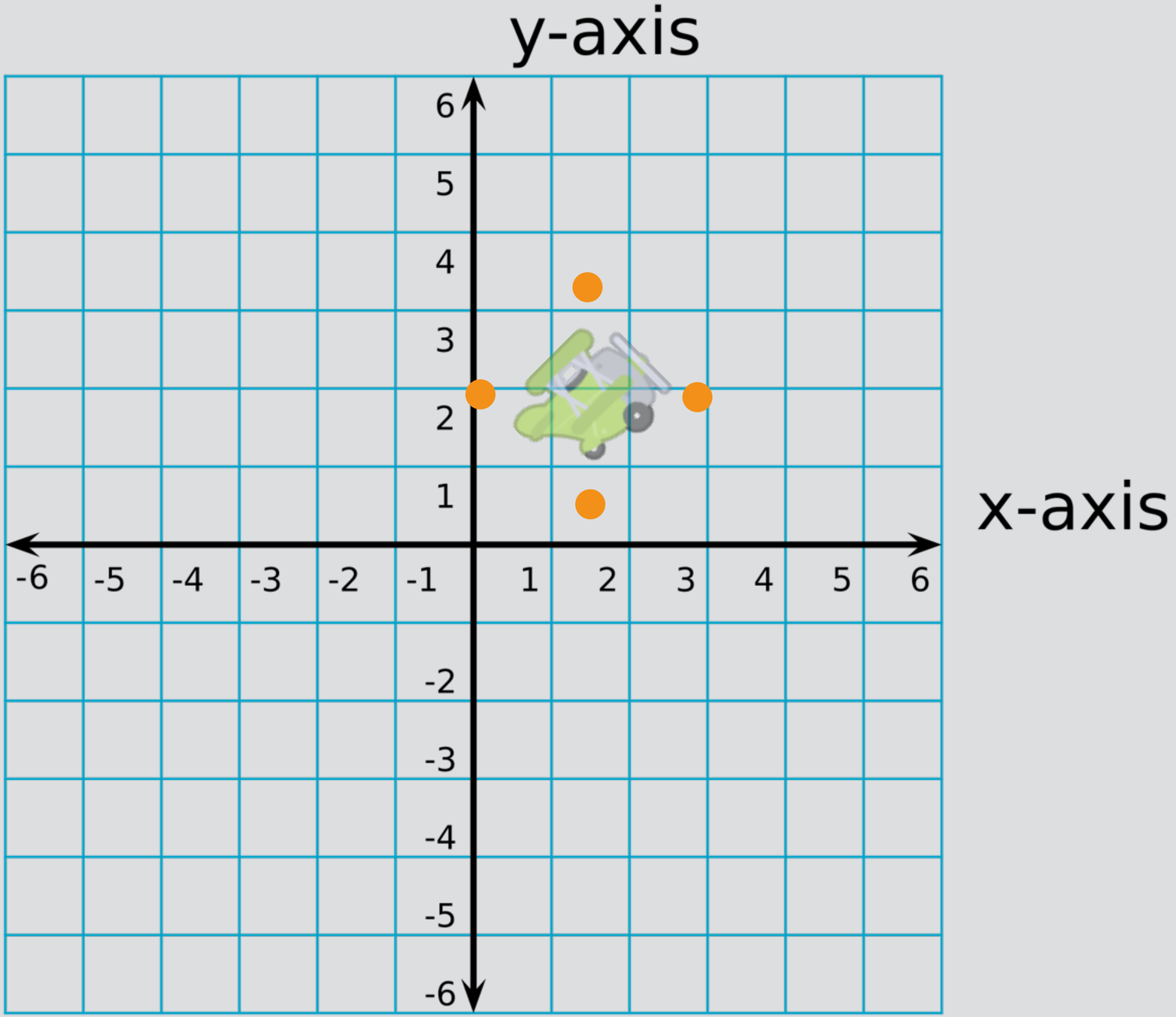

y-axis
x-axis
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6

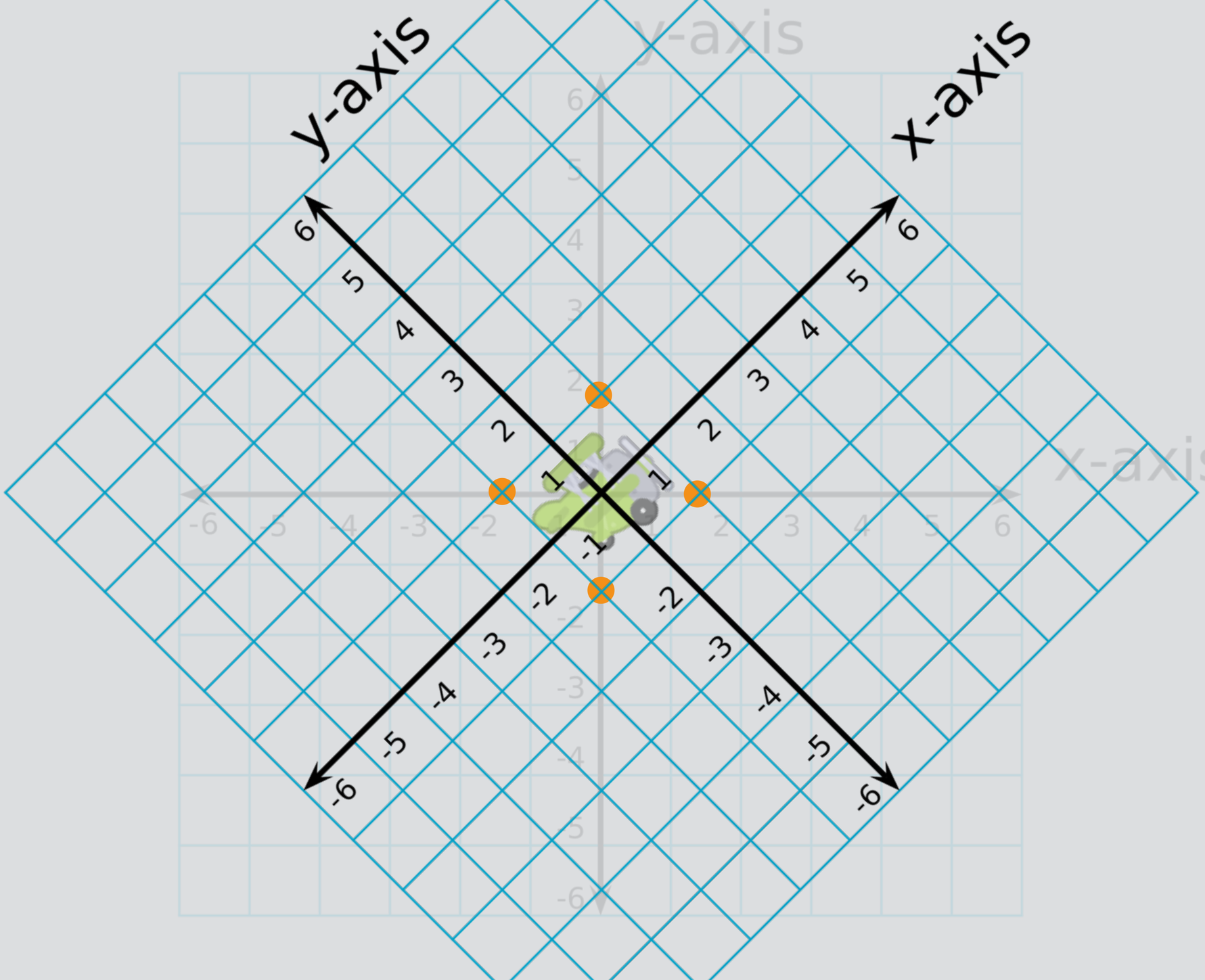

y-axis
x-axis
6
5
4
3
2
1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6

```
Matrix modelMatrix;
modelMatrix.Rotate(45.0 * (3.1415926 / 180.0));
modelMatrix.Translate(2.0, 0.0, 0.0);
```


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


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


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


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