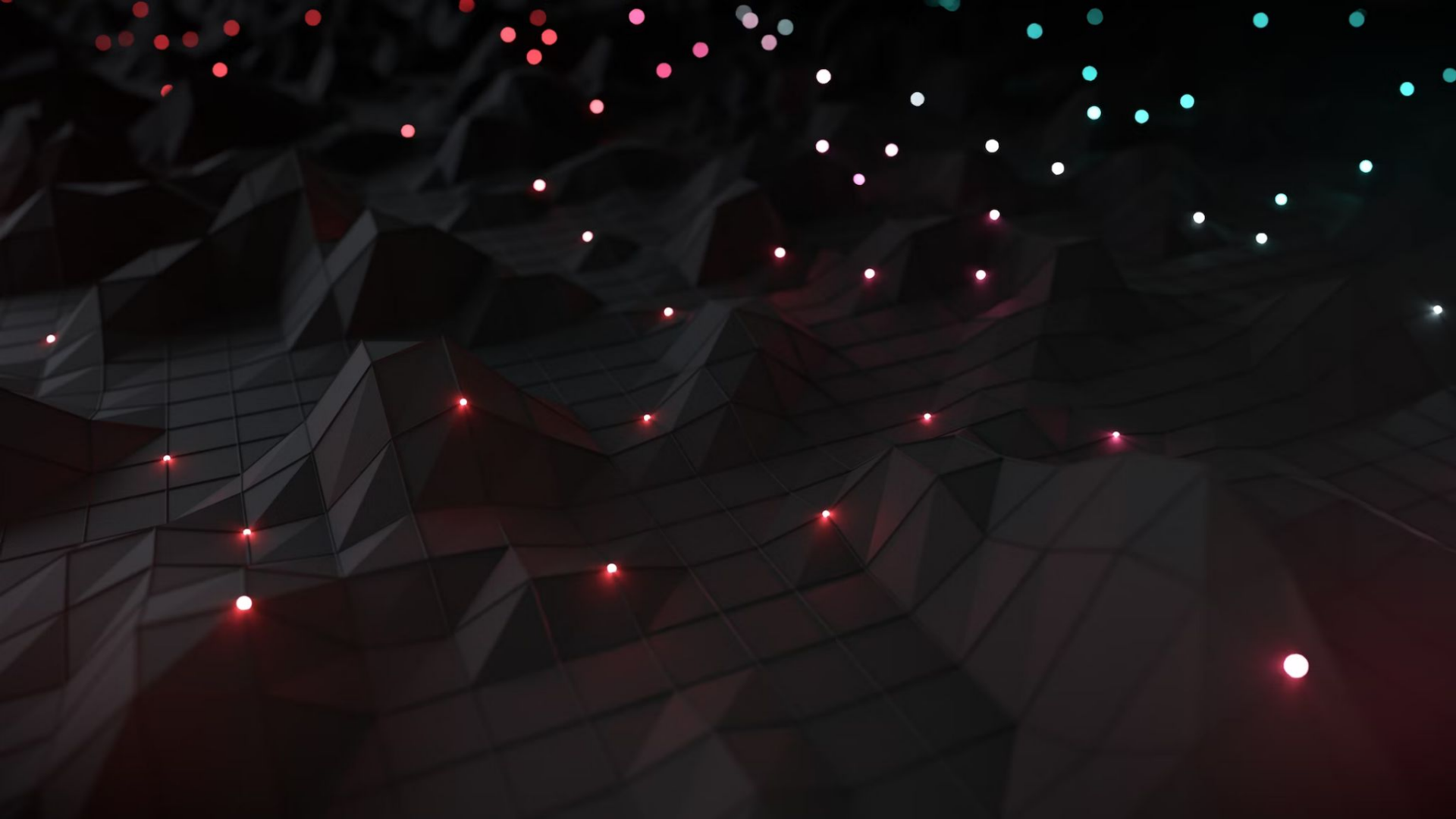
Flame & *(real)* 3D

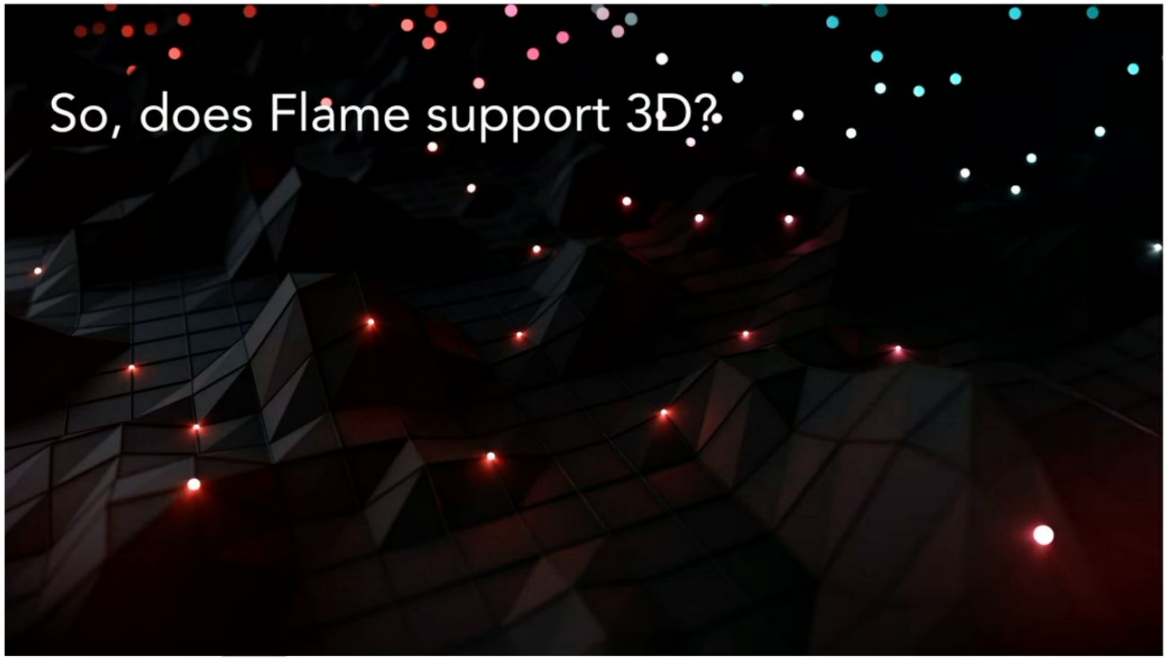
or "Can we make 3D games with Flame?"

Luan Nico

FLAME



Flame and (pseudo) 3d



Luan
Nico

So, does Flame support 3D?

- What do you mean by "Flame"?
- What do you mean by "support"?
- What do you mean by "3D"?

https://docs.flutter.dev/resources/faq

Can I run Dart code in the background of an Flutter app?

Yes, you can run Dart code in a background process on both iOS and Android For more information, see the free Medium article Executing Dart in the Background with Flutter Plugins and Geofencing.

Can I use JSON/XML/protobuffers, etc. with Flutter?

Absolutely. There are libraries on pub.dev for JSON, XML, protobufs, and many other utilities and formats.

For a detailed writeup on using JSON with Flutter, check out the JSON tutorial.

Can I build 3D (OpenGL) apps with Flutter?

Today we don't support for 3D via OpenGL ES or similar. We have long-term plans to expose an optimized 3D API, but right now we're focused on 2D.

natively

Why is my APK or IPA so big?

Usually, assets including images, sound files, fonts, etc, are the bulk of an APK or IPA. Various tools in the Android and iOS ecosystems can help you understand what's inside of your APK or IPA.

Also, be sure to create a *release build* of your APK or IPA with the Flutter tools. A release build is usually *much* smaller than a *debug build*.

Learn more about creating a release build of your Android app, and creating a release build of your iOS app. Also, check out Measuring your app's size.

Getting started with Flutter GPU

Build custom renderers and render 3D scenes in Flutter.

Brandon DeRosier · Follow

Published in Flutter · 17 min read · Aug 6, 2024

The (*Real*) 3D

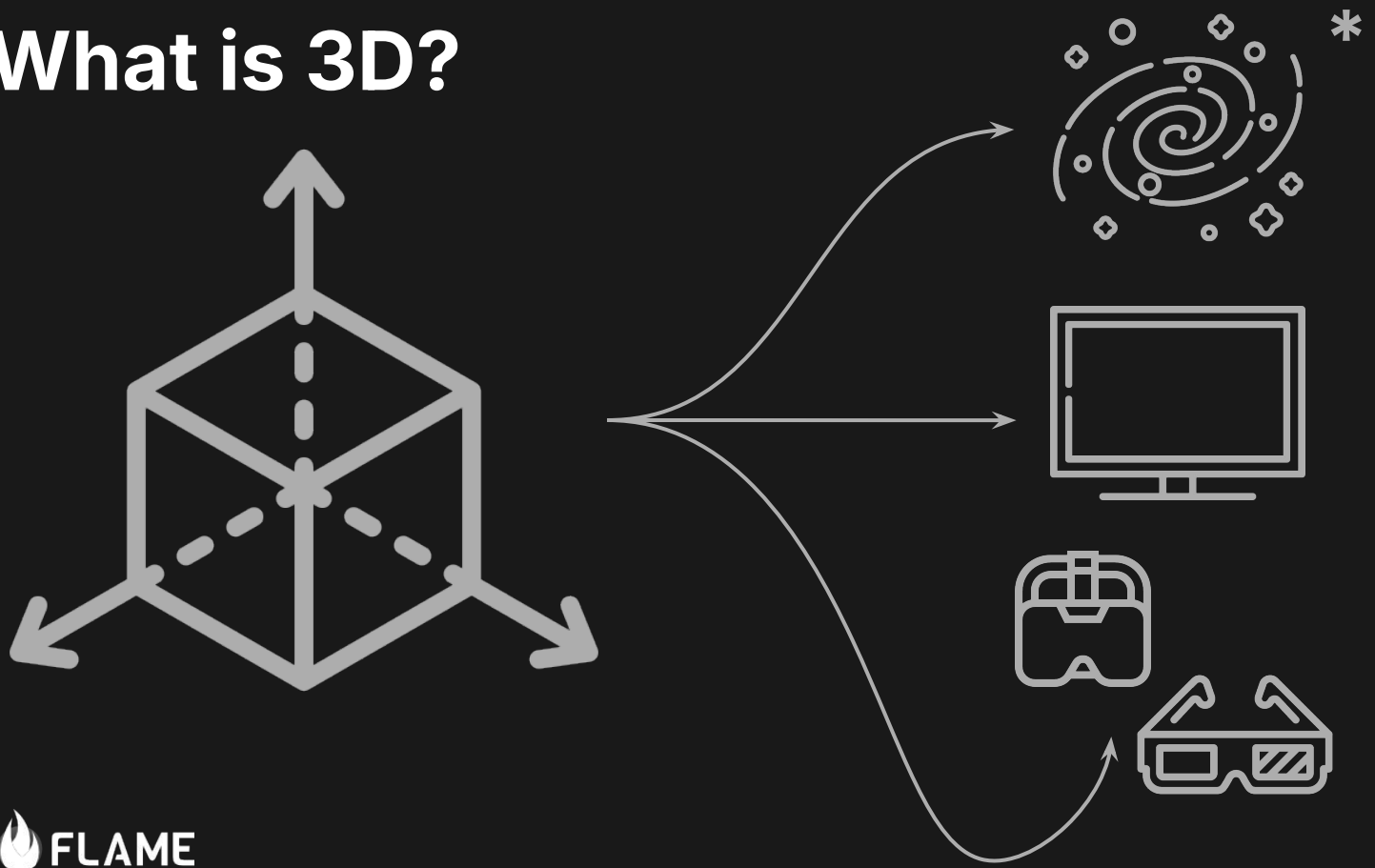
FLAME

The (*Real*) 3D?

3D?

FLAME

What is 3D?



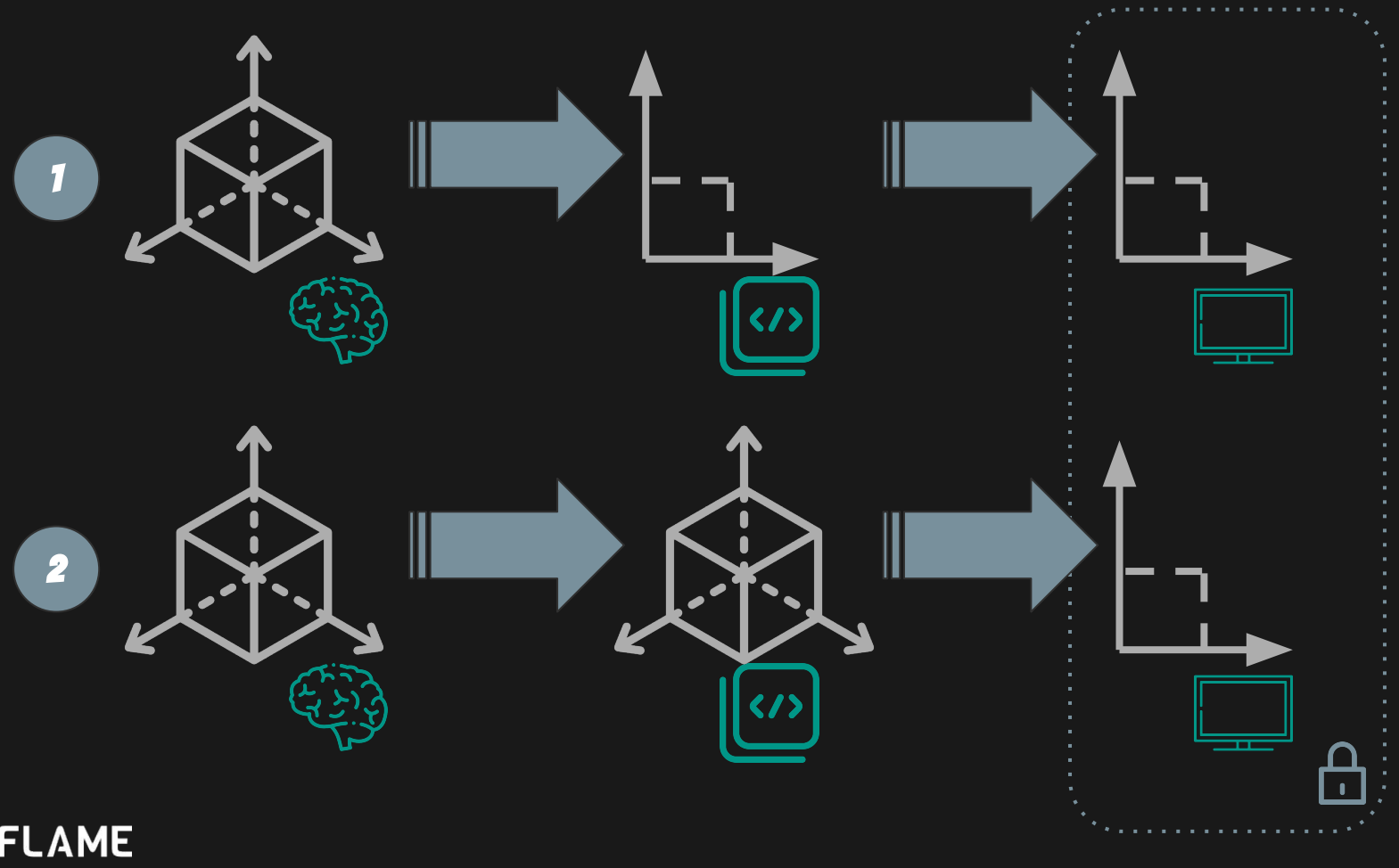
FLAME

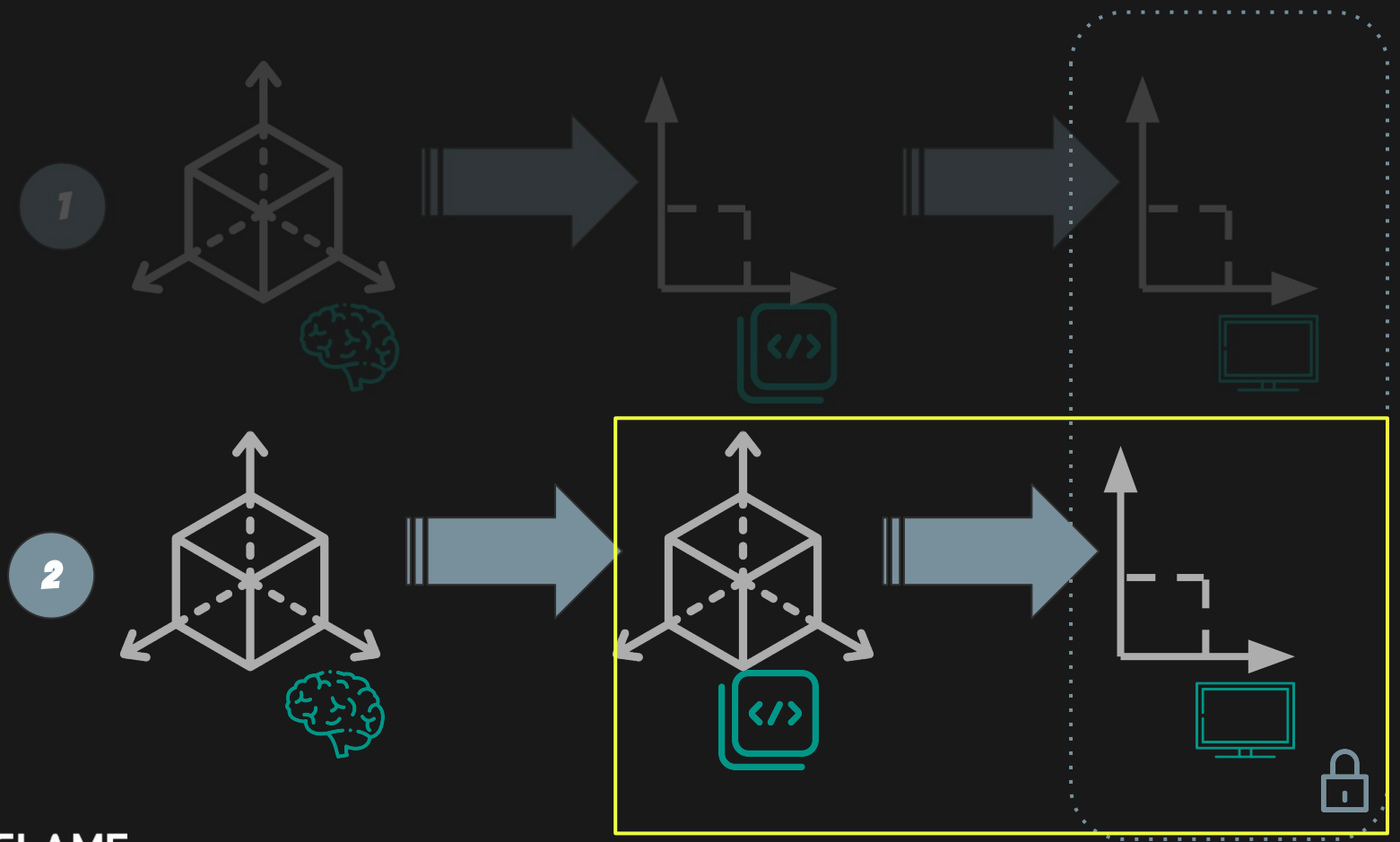
FLAME

The (*Real*) 3D?

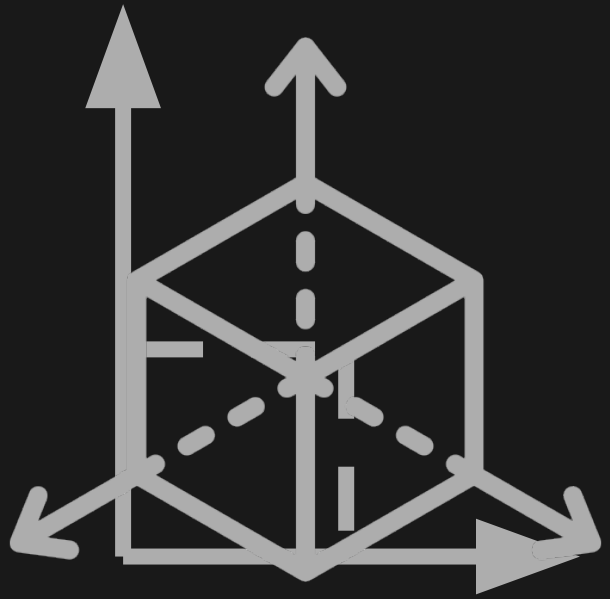
3D rendering?

FLAME


1
2
FLAME


1
2
FLAME

3D World Representation - Basics

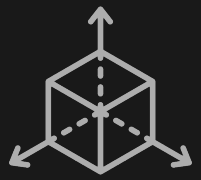


```
class Player {
    Vector3 position;
}
```

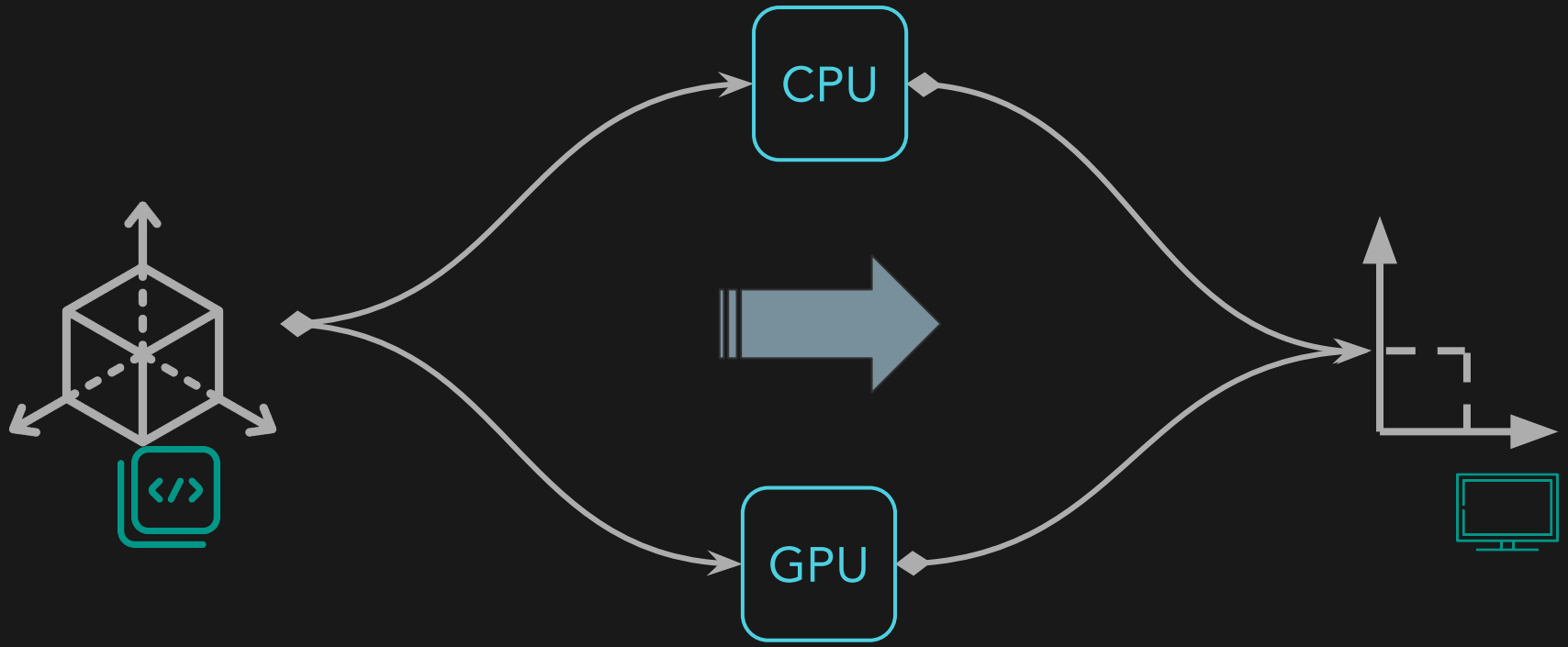
FLAME

The (*Real*) 3D?

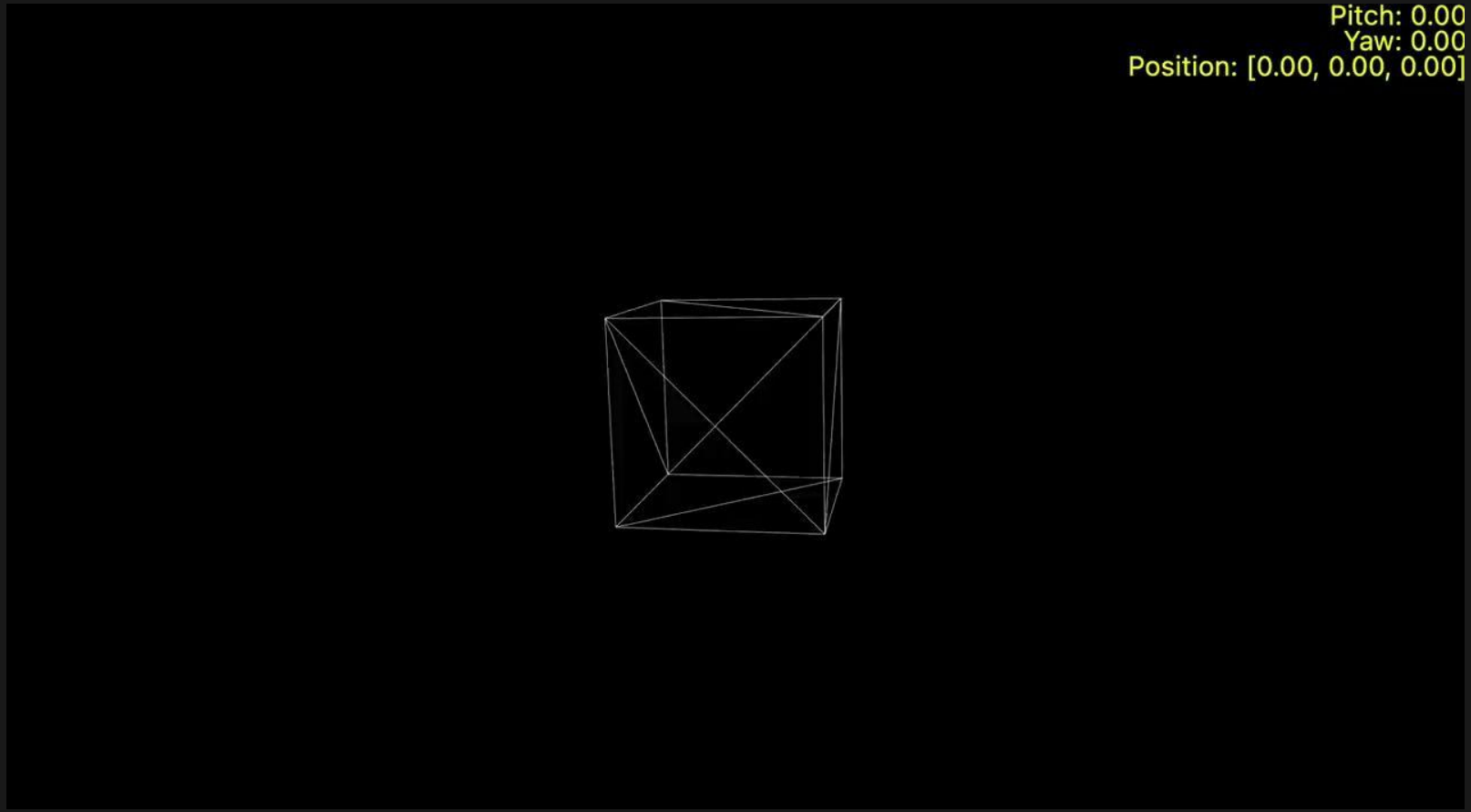
3D-world rendering?



FLAME

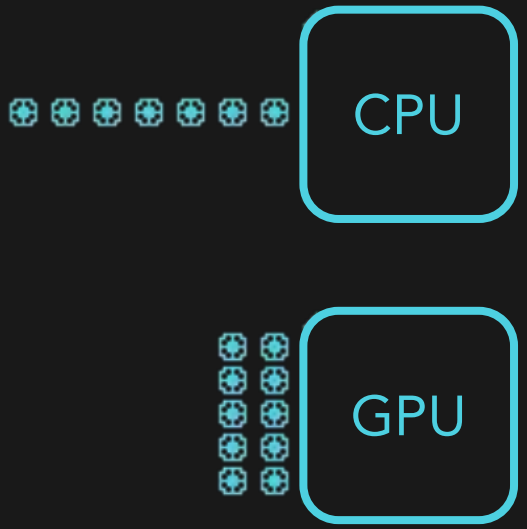


FLAME



FLAME

github.com/luanpotter/flame_3d_testbed



FLAME

OpenGL®

OpenCL™

Vulkan®

OpenACC

More Science, Less Programming

GPU

NVIDIA

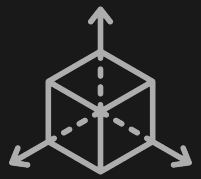
CUDA

FLAME

The (*Real*) 3D?

gpu-accelerated 3D-world rendering

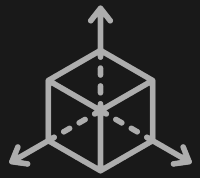
GPU



FLAME

The (*Real*) 3D?

first-class gpu-accelerated 3D-world rendering



FLAME

Can we do it?

Yes.******

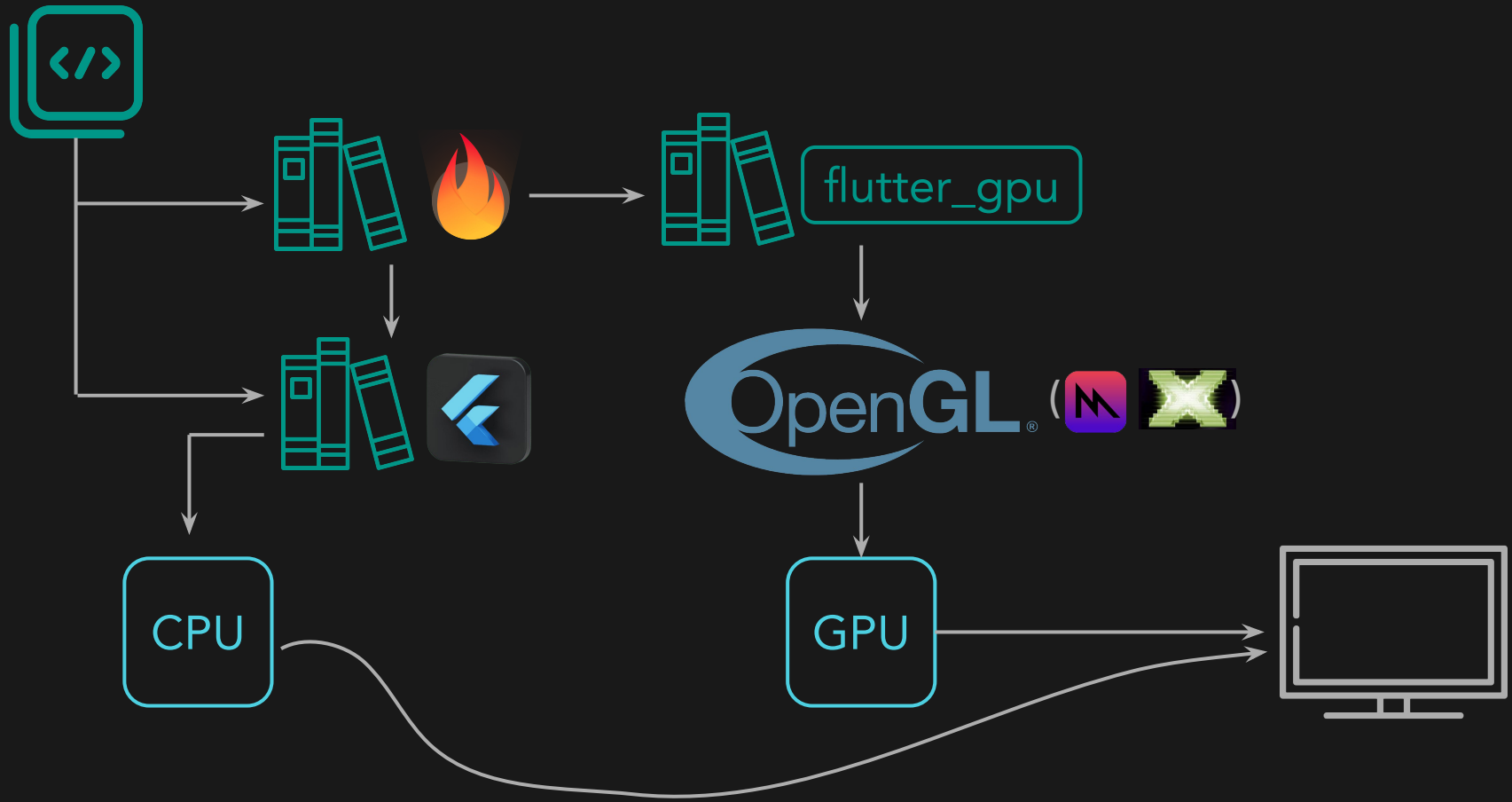
FLAME

flame_3d
flame_3d branch

flame_3d_extras
+ playground

```yaml
dependency_overrides:
  flame_3d:
    path: ../flame/packages/flame_3d
  flame_3d_extras:
    path: ../flame_3d_extras
```

FLAME

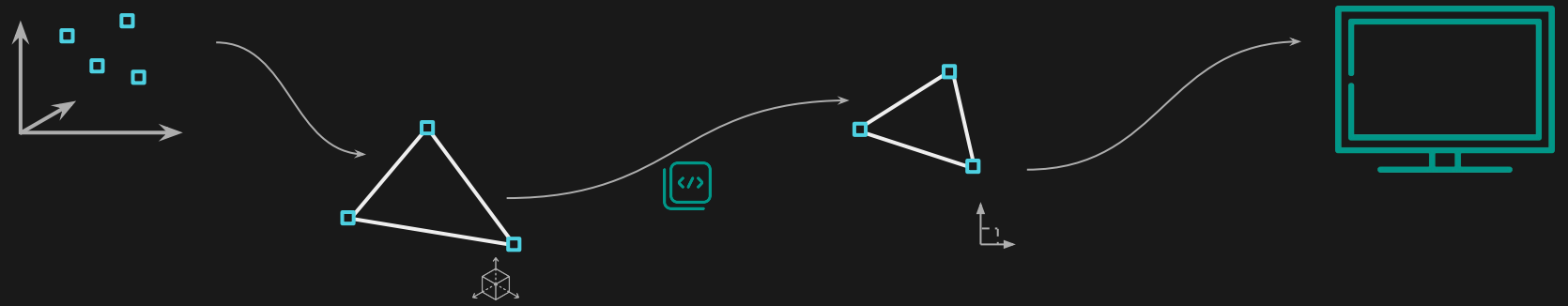


FLAME

3D Rendering 101

(with flame_3d)

3D Projection - The Secret



FLAME


always has been
wait so it is
all matrices???
$\begin{pmatrix} S_x & 0 & 0 & T_x \\ 0 & S_y & 0 & T_y \\ 0 & 0 & S_z & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$
imgflip.com

FLAME

3D Projections - Transformations - TRS

$$M = \begin{pmatrix} S_x & 0 & 0 & T_x \\ 0 & S_y & 0 & T_y \\ 0 & 0 & S_z & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Vector3 translation;

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) & 0 \\ 0 & -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & -sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

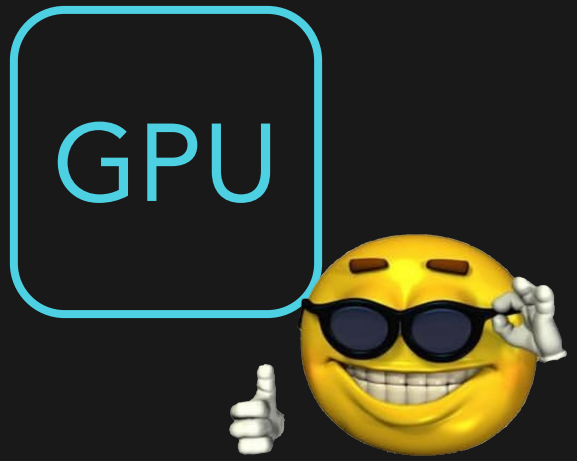
$$R_x(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Quaternion rotation;

$$T = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

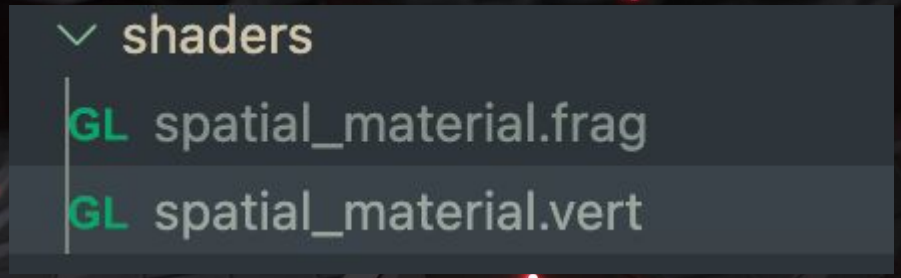
Vector3 scale;

FLAME



FLAME

Shaders: The Key




```
~/projects/gamedev/flame/packages/flame_3d
dart ./bin/build_shaders.dart
```

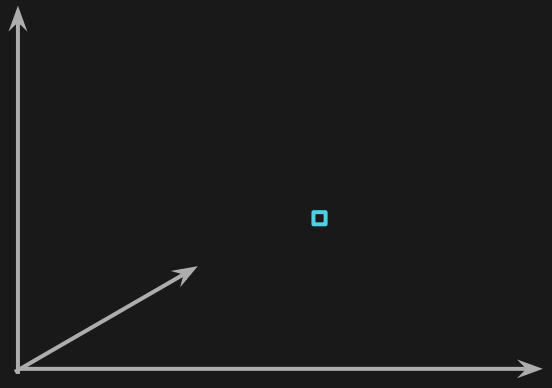
https://learnopengl.com

shaders > GL spatial_material.vert > ...

```
#version 460 core

in vec3 vertexPosition;
in vec2 vertexTexCoord;
in vec4 vertexColor;
in vec3 vertexNormal;

out vec2 fragTexCoord;
out vec4 fragColor;
out vec3 fragPosition;
out vec3 fragNormal;

uniform VertexInfo {
  mat4 model;
  mat4 view;
  mat4 projection;
} vertex_info;

void main() {
  // Calculate the modelview projection matrix
  mat4 modelViewProjection = vertex_info.projection * vertex_info.view * vertex_info.model;

  // Transform the vertex position
  gl_Position = modelViewProjection * vec4(v0: vertexPosition, v1: 1.0);

  // Pass the interpolated values to the fragment shader
  fragTexCoord = vertexTexCoord;
  fragColor = vertexColor;

  // Calculate the world-space position and normal
  fragPosition = vec3(value: vertex_info.model * vec4(v0: vertexPosition, v1: 1.0));
  fragNormal = mat3(transpose(inverse(vertex_info.model))) * vertexNormal;
}
```

Vertices

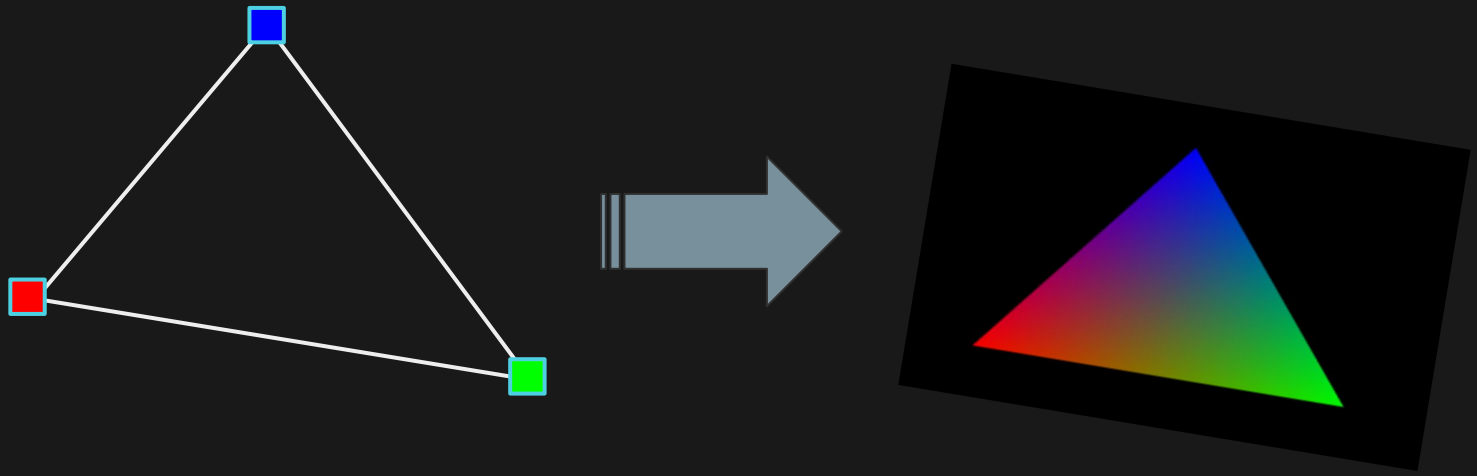
3D Projection - Vertex



- Position
- Color / Texture Information
- Normal Information
- Joint Information

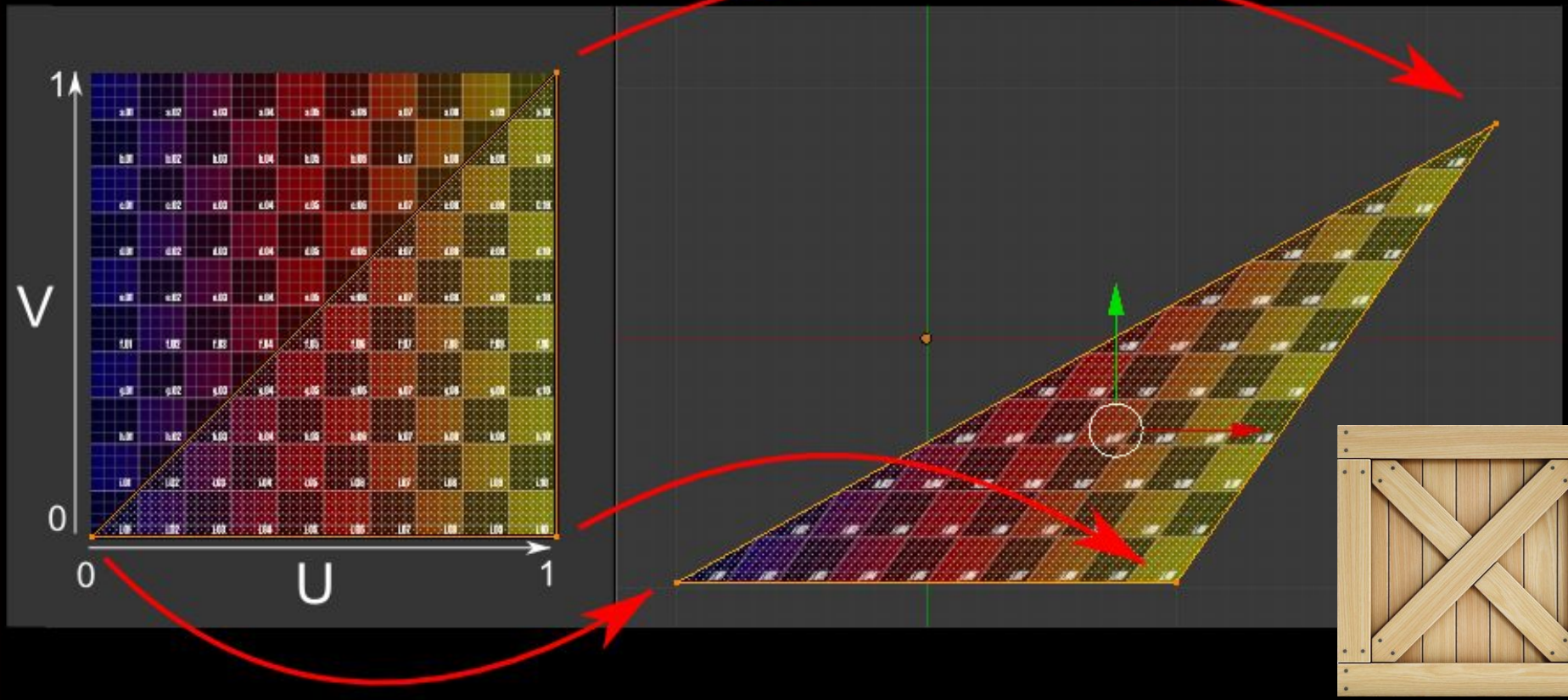
FLAME

Color



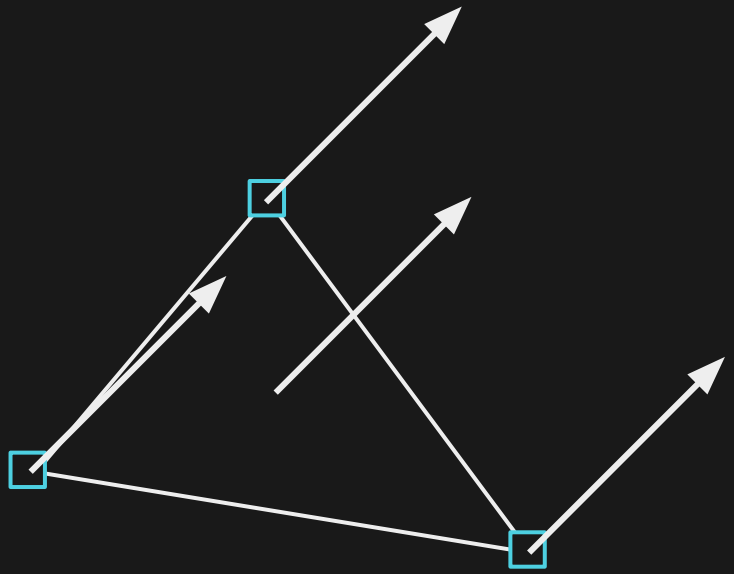
FLAME

Textures



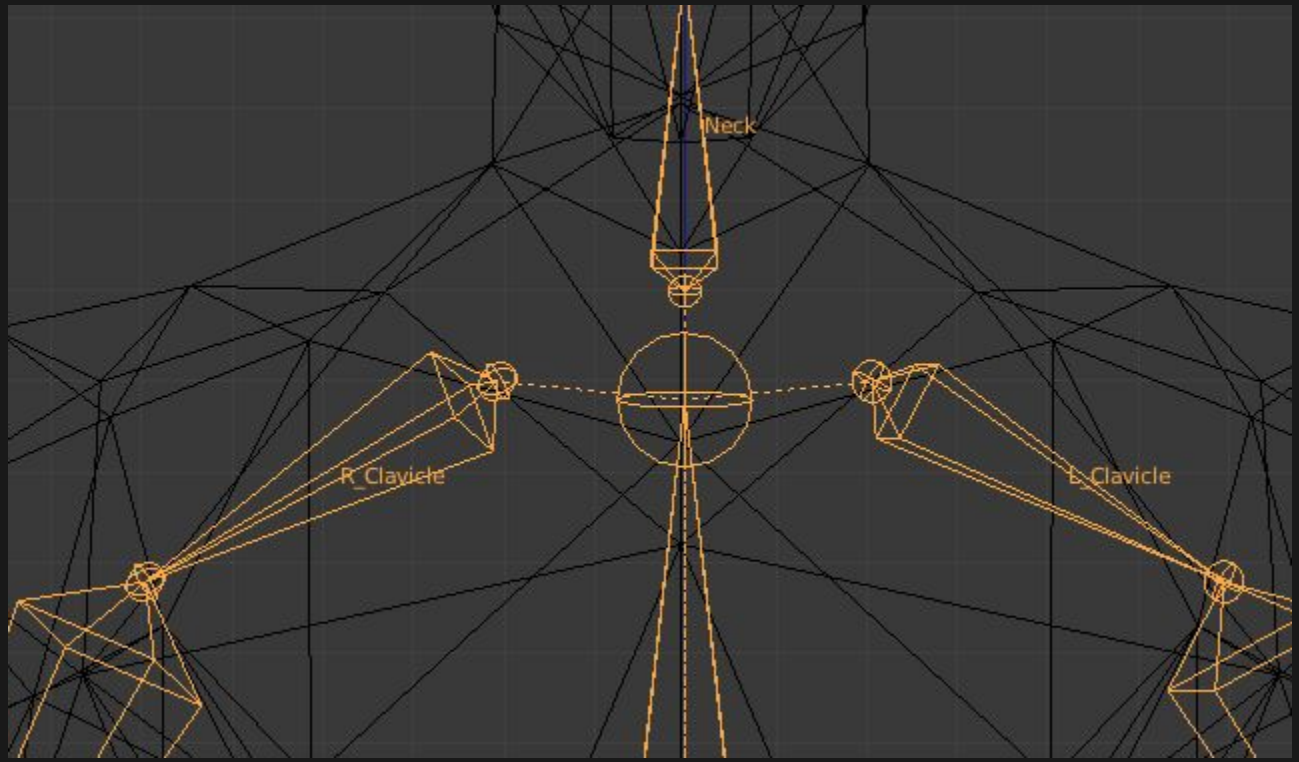
https://webglfundamentals.org/webgl/lessons/webgl-3d-textures.html

Normals



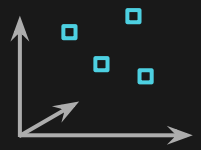
FLAME

Joints & Weights (for Skeletal Animation)



FLAME

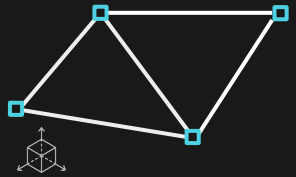
Vertex



```
class Vertex {
  Vertex({
    required Vector3 position,
    required Vector2 texCoord,
    required Color color,
    Vector3? normal,
    Vector4? joints,
    Vector4? weights,
  }) {
    // ...
    this.storage = ...
  }
}
```

FLAME

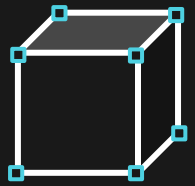
Surface (Triangle Strip)



```dart
class Surface {
  Surface({
    required List<Vertex> vertices,
    required List<int> indices,
    Material? material;
    Map<int, int>? jointMap;
    /**
     * If `true`, the normals will be
     * calculated if not provided.
     */
    bool calculateNormals = true,
  }) {
    // ...
  }
}
```

FLAME

Mesh - Solid

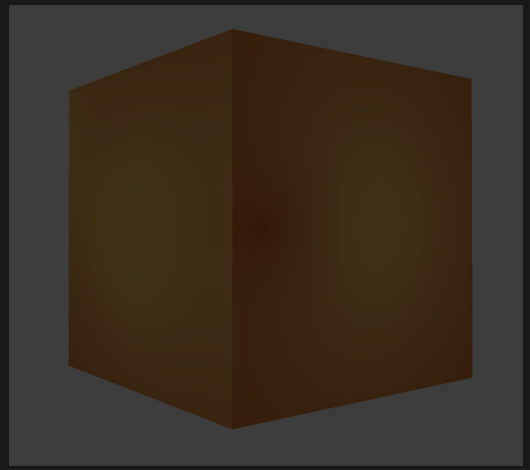


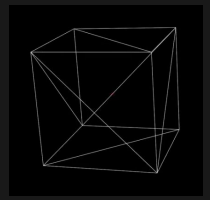
```
class Mesh {
  final List<Surface> surfaces;
  Aabb3 get aabb;
}
```

FLAME

Simple Cube

```
await add(
  MeshComponent(
    mesh: CuboidMesh(
      size: Vector3.all(8.0),
    ),
  ),
);
```





```
final Vector3(:x, :y, :z) = size / 2;

final vertices = [
  // Face 1 (front)
  vertex(
    position: Vector3(-x, -y, -z),
    texCoord: Vector2(0, 0),
    normal: Vector3(0, 0, -1),
  ),
  vertex(
    position: Vector3(x, -y, -z),
    texCoord: Vector2(1, 0),
    normal: Vector3(0, 0, -1),
  ),
  vertex(
    position: Vector3(x, y, -z),
    texCoord: Vector2(1, 1),
    normal: Vector3(0, 0, -1),
  ),
  vertex(
    position: Vector3(-x, y, -z),
    texCoord: Vector2(0, 1),
    normal: Vector3(0, 0, -1),
  ),

  // Face 2 (back)
  vertex(
    position: Vector3(-x, -y, z),
    texCoord: Vector2(0, 0),
    normal: Vector3(0, 0, 1),
  ),
  vertex(
    position: Vector3(x, -y, z),
    texCoord: Vector2(1, 0),
    normal: Vector3(0, 0, 1),
  ),
  vertex(
    position: Vector3(x, y, z),
    texCoord: Vector2(1, 1),
    normal: Vector3(0, 0, 1),
  ),
  vertex(
    position: Vector3(-x, y, z),
    texCoord: Vector2(0, 1),
    normal: Vector3(0, 0, 1),
```

FLAME

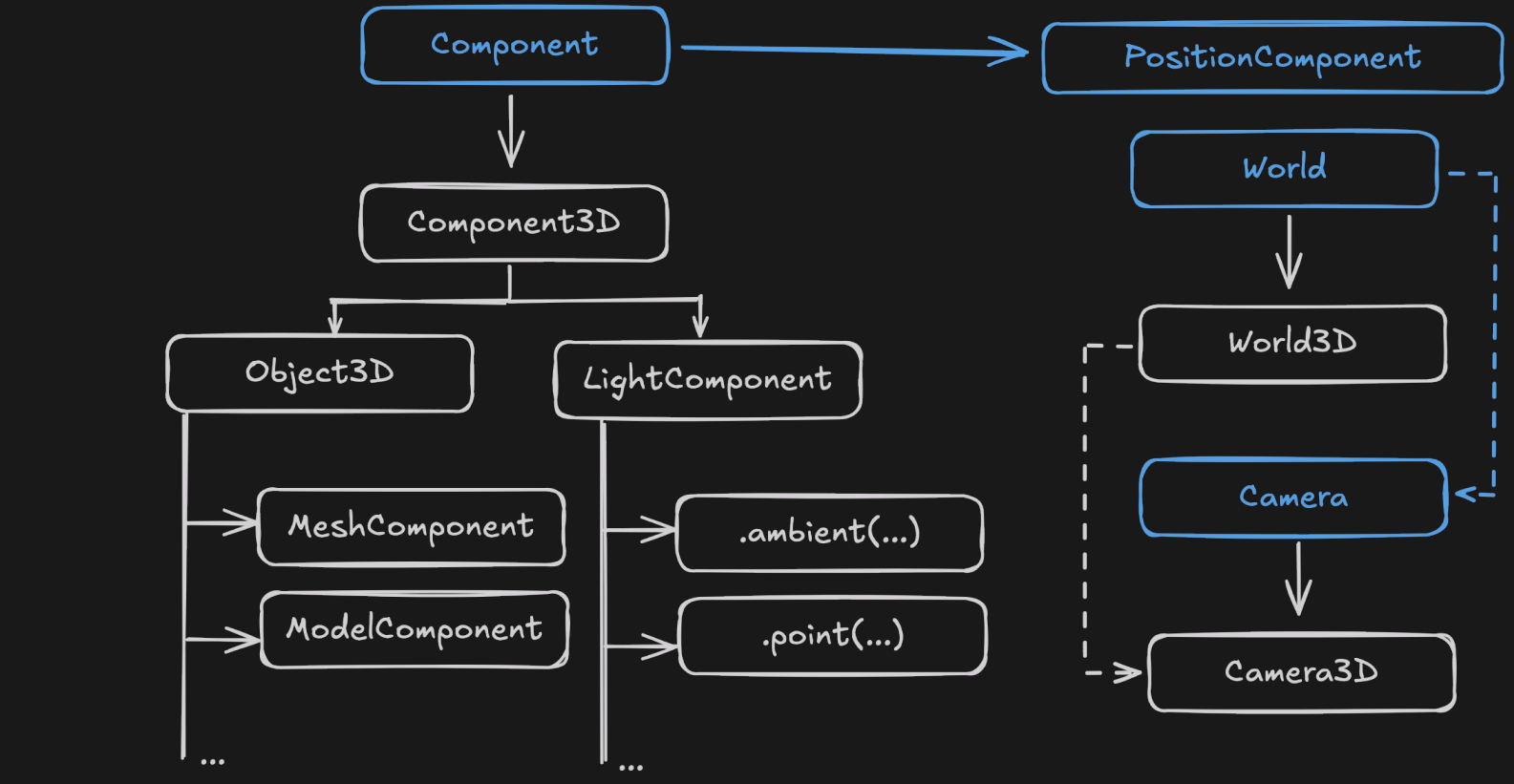
Lighting

FLAME

LightComponent

```
await add(
  LightComponent.ambient(
    color: const Color(0xFFFFFFFF),
    intensity: 0.75,
  ),

  LightComponent.point(
    position: Vector3(...),
    color: ...,
    intensity: 20.0,
  ),
  // ...
);
```

FLAME


Component
PositionComponent
Component3D
World
Object3D
LightComponent
World3D
MeshComponent
.ambient(...)
Camera
ModelComponent
.point(...)
Camera3D
...
...

FLAME

The Model Guides Us!

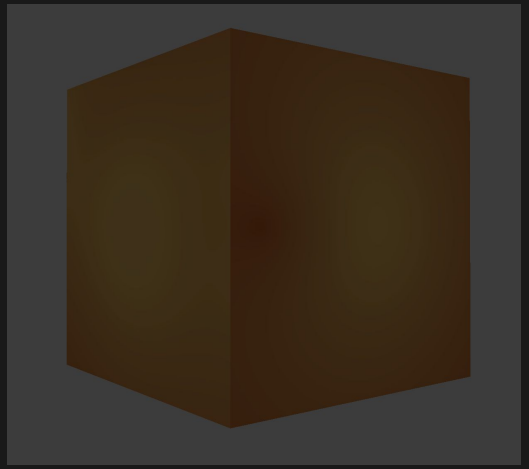
- PBR - Physics Based Rendering

$$L_o(p, \omega_o) = \int_{\Omega} (k_d \frac{c}{\pi} + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p, \omega_i) n \cdot \omega_i d\omega_i$$

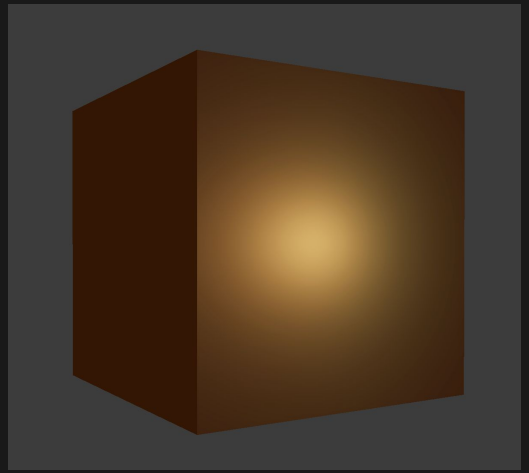
- Arbitrary number of lights
- Other types of lights
- Shadows!

FLAME

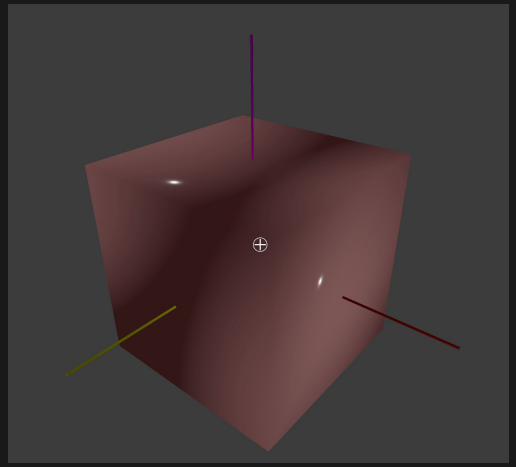
Face vs Vertex Normals



vertex normals

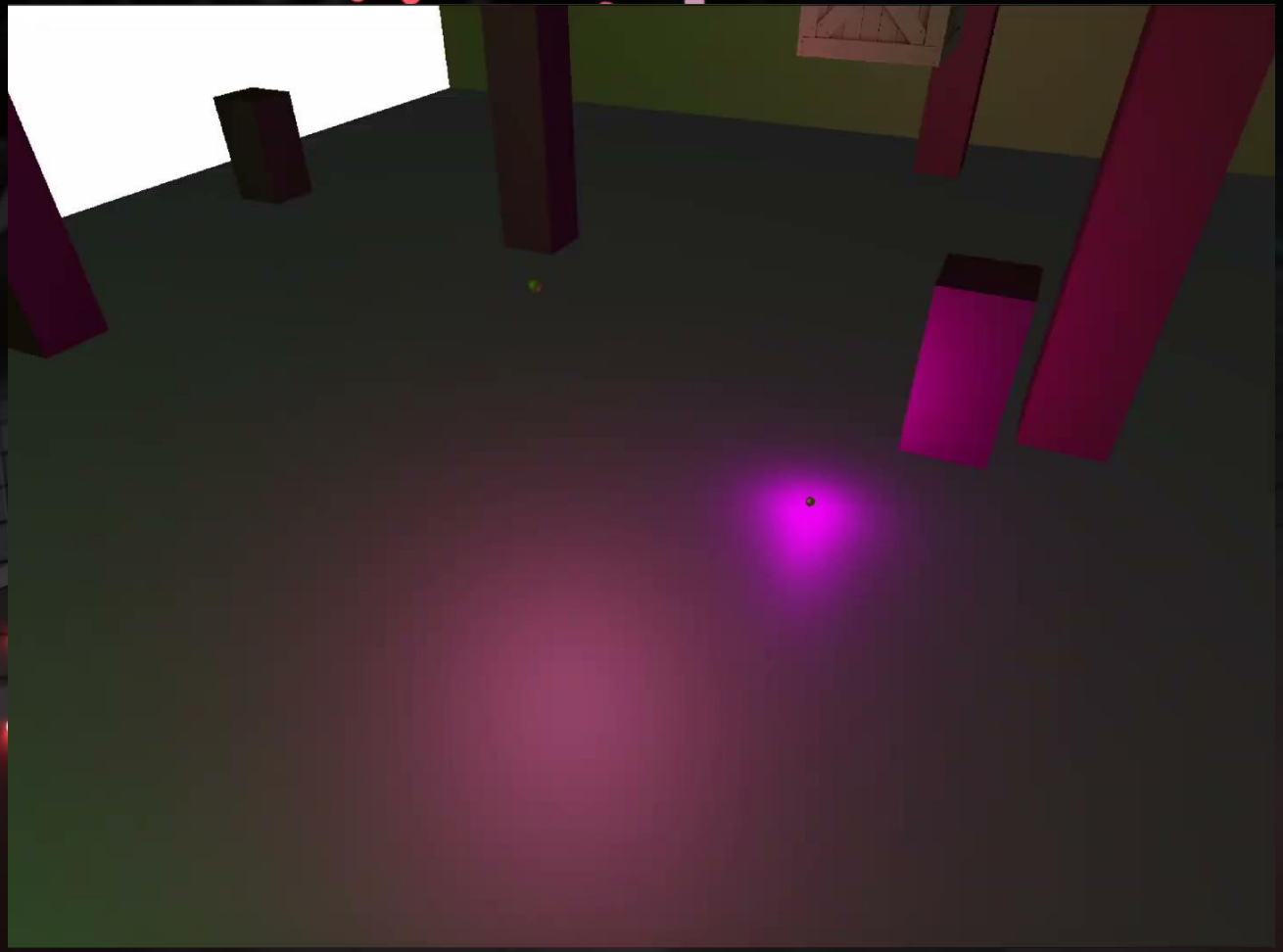


face normals



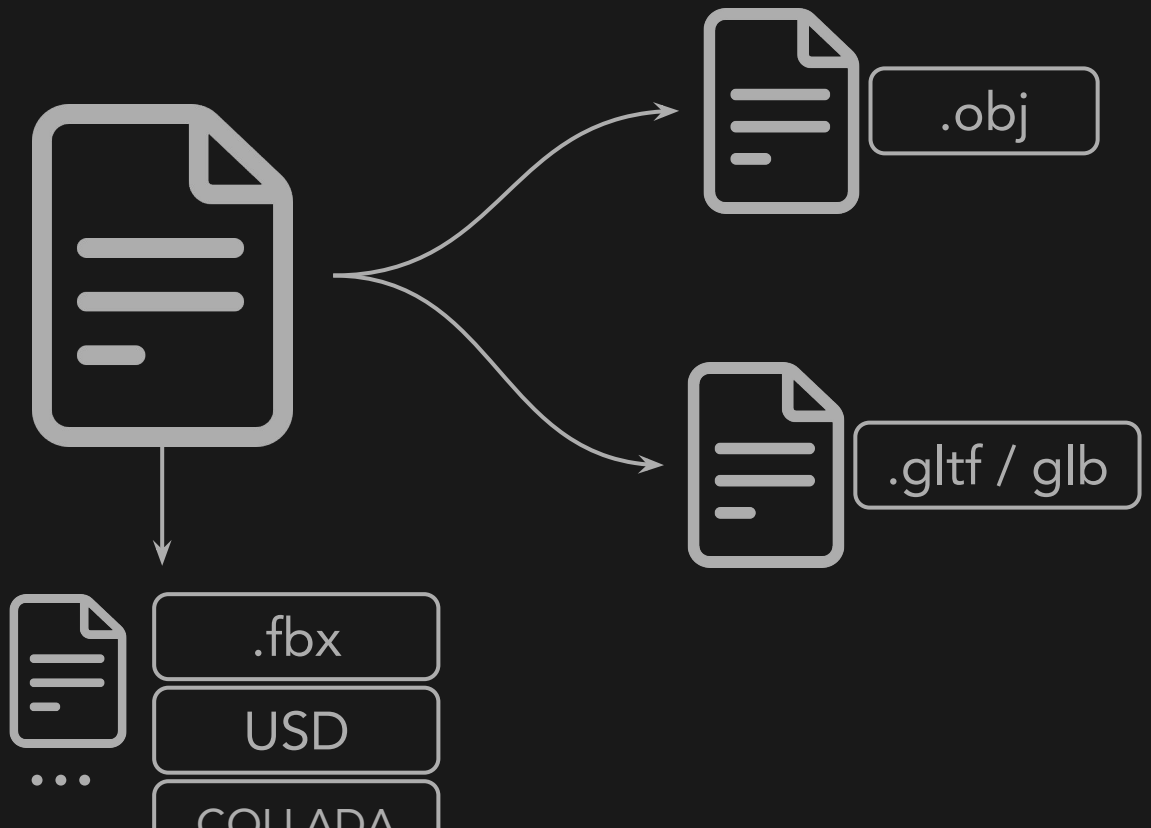
gltf model

FLAME



Parsing Models

FLAME



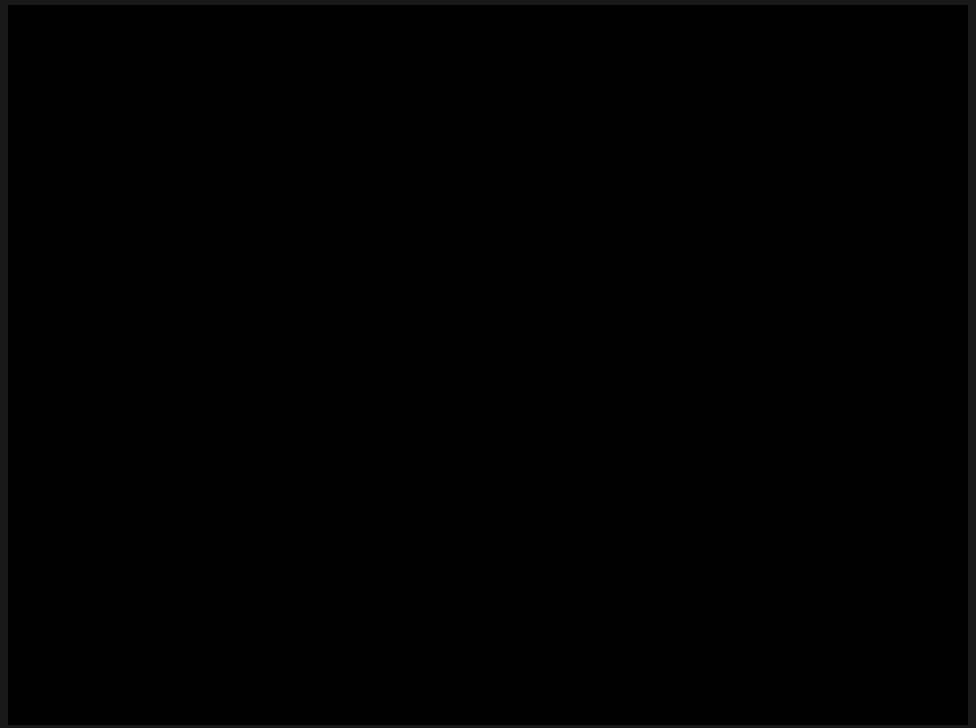
FLAME

.obj file

```
# vertex (x y z)
v -0.000752473 0.044662 0.160067
# normal (x y z)
vn -0.146317 0.97078 0.194263
# texel (u v)
vt 0.123 0.456
# material
usemtl mat0
# surface (v/vt/vn)
f 1/1/1 2/1/1 3/1/1
```

FLAME

Defend the Donut



FLAME

.gltf / .glb files

- byte buffers!
- scenes, nodes, meshes
- textures
- animation
- skin / skeletal animation

FLAME

Model

```dart
class Model {
  final Map<int, ModelNode> nodes;
  final Map<String, ModelAnimation> animations;
  // ...
}

class ModelNode {
  final int nodeIndex;
  final int? parentNodeIndex;
  final Matrix4 transform;
  final Mesh? mesh;
  final Map<int, ModelJoint> joints;
  // ...
}

class ModelAnimation {
  final String name;
  final Map<int, NodeAnimation> nodes;
  final double _lastTime;
  double _clock = 0;
  // ...
}
```

FLAME

ModelComponent

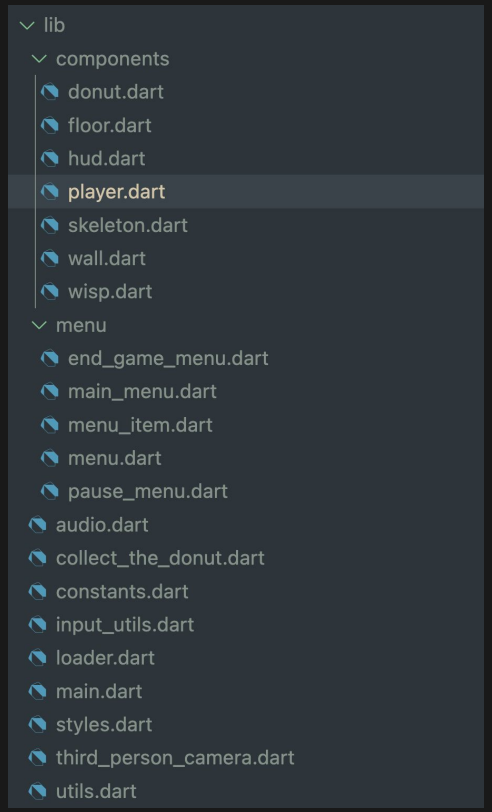
```
await add(
    player = ModelComponent(
        model: await ModelParser.parse('objects/rogue.glb'),
        position: ..., // scale, rotation, transform
    ),
);

// later...

player.playAnimationByName('Walking_B');
player.stopAnimation();
player.hideNodeByName('Knife');
```

But... An Actual Game?

FLAME

Collect the Donut

github.com/luanpotter/collect_the_donut



FLAME


COLLECT THE
DONUT
- start -

What to expect?

- Polish (flame_3d branch, flame_3d_extras)
- SSBO (not even simple arrays)
- Shadows
- Physics
- Optimization
- A lot more…

```
uniform JointMatrices {
  mat4 joint0;
  mat4 joint1;
  mat4 joint2;
  mat4 joint3;
  mat4 joint4;
  mat4 joint5;
  mat4 joint6;
  mat4 joint7;
  mat4 joint8;
  mat4 joint9;
  mat4 joint10;
  mat4 joint11;
  mat4 joint12;
  mat4 joint13;
  mat4 joint14;
  mat4 joint15;
} joints;

mat4 jointMat(float jointIndex) {
  if (jointIndex == 0.0) {
    return joints.joint0;
  } else if (jointIndex == 1.0) {
    return joints.joint1;
  } else if (jointIndex == 2.0) {
    return joints.joint2;
  } else if (jointIndex == 3.0) {
    return joints.joint3;
  } else if (jointIndex == 4.0) {
    return joints.joint4;
  } else if (jointIndex == 5.0) {
    return joints.joint5;
  } else if (jointIndex == 6.0) {
    return joints.joint6;
  } else if (jointIndex == 7.0) {
    return joints.joint7;
  } else if (jointIndex == 8.0) {
    return joints.joint8;
  } else if (jointIndex == 9.0) {
    return joints.joint9;
  } else if (jointIndex == 10.0) {
    return joints.joint10;
  } else if (jointIndex == 11.0) {
    return joints.joint11;
  } else if (jointIndex == 12.0) {
    return joints.joint12;
```

FLAME

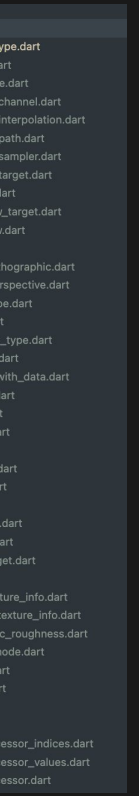
Some Fun Learnings

And why we still need to refine flame_3d

FLAME

Data Model is Everything!

- GLTF as a "blueprint"
- typed data structure from the docs
- sample models from their repo



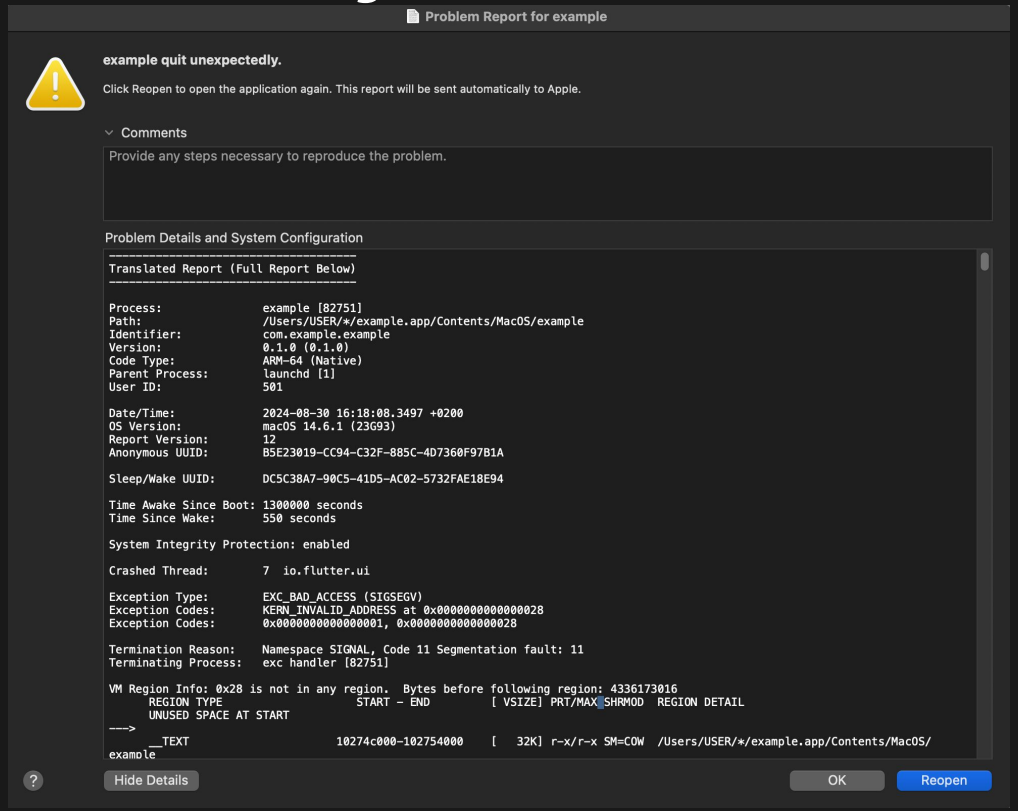
FLAME

FLAME

Shaders are Devilish

- Type-highly-unsafe
- "Optimize" the "fun" out of debugging
  - return magenta; trick
- Manual byte wiring

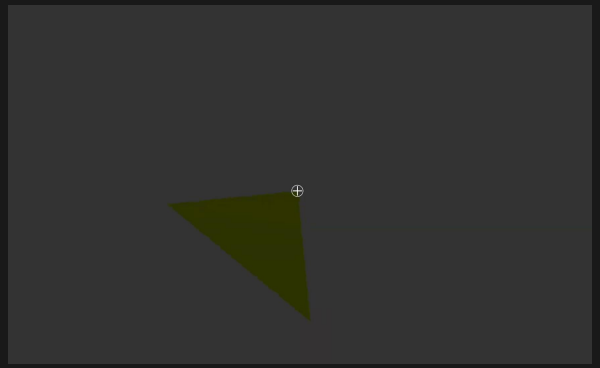
Cryptic Errors - Bytes Galore



FLAME

Quaternion Interpolation

```
static Quaternion lerp(
  Quaternion q0,
  Quaternion q1,
  double t,
) {
  return q0 + (q1 - q0).scaled(t);
}
```



FLAME

```dart
/// Some background on the correct shortest-path implementation can be found
/// here:
/// https://blog.magnum.graphics/backstage/the-unnecessarily-short-ways-to-do-a-quaternion-slerp/
static Quaternion _slerp(
  Quaternion q0,
  Quaternion q1,
  double t, {
  double epsilon = 10e-3,
}) {
  if (isEqual(q0, q1)) {
    return q0;
  }

  final dot = q0.dot(q1);
  if (1 - dot < epsilon) {
    // The quaternions are very close, so the linear interpolation algorithm
    // will be a good approximation.
    // This will prevent NaN from the slerp algorithm.
    return lerp(q0, q1, t);
  }

  final angle = acos(dot.abs());

  final q1Prime = dot >= 0 ? q1 : q1.scaled(-1);
  final a = sin((1 - t) * angle) / sin(angle);
  final b = sin(t * angle) / sin(angle);

  return q0.scaled(a) + q1Prime.scaled(b);
}
```

Order Matters

FLAME

Thank you!

Questions?

FLAME


FLAME

- THE END -

FLAME

Credits

https://romannurik.github.io/SlidesCodeHighlighter/?theme=flutter-interact-19&font=Source+Code+Pro&tab=4&size=40&sel=focus

itch.io assets

flaticon icons

Renan for the slide deck template

unsplash for the photos

https://butterymilk.itch.io/tiny-wonder-farm-asset-pack

FLAME

Recommended Reading

https://www.brainvoyager.com/bv/doc/UsersGuide/CoordsAndTransforms/SpatialTransformationMatrices.html

https://www.3dgep.com/understanding-the-view-matrix/

https://www.youtube.com/watch?v=ih20l3pJoeU

https://www.scratchapixel.com/lessons/3d-basic-rendering/perspective-and-orthographic-projection-matrix/building-basic-perspective-projection-matrix.html

https://lodev.org/cgtutor/raycasting.html

FLAME

Color Palette


#FFFFFF


#212121


#303030


#ADADAD


#009688


#EEEEEE


#78909C


#FFAB40


#4DD0E1


#EEFF41


#191919

FLAME

More slides in case people ask some questions

42

simple

modular

code first

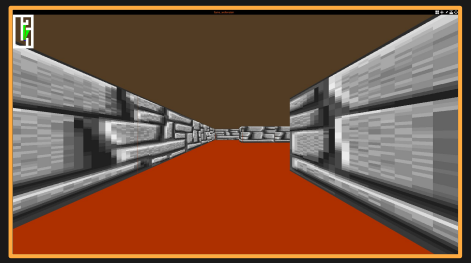
open

hackable

FLAME

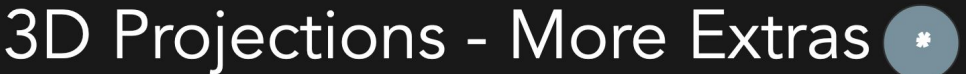
FLAME

VS



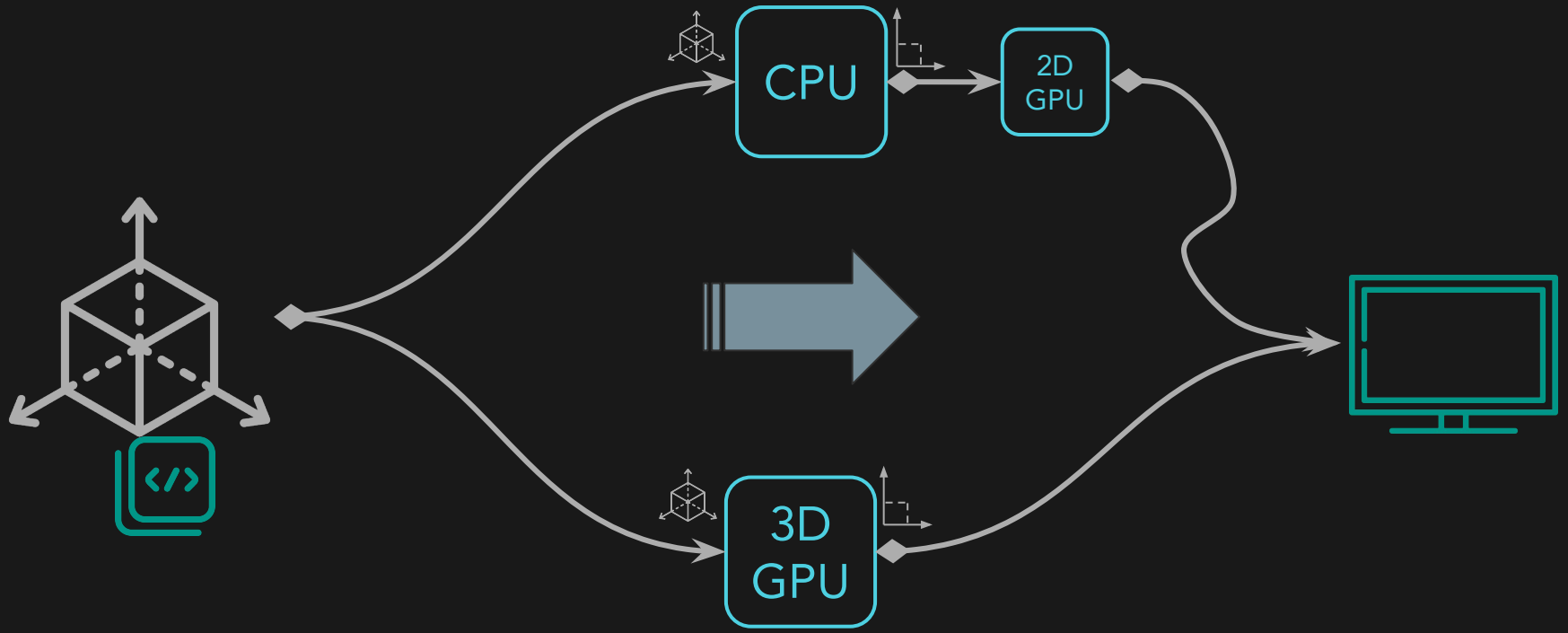
FLAME

3D Projections - More Extras



- Culling
- z-buffer
- Lights
- Textures
- Physics
- Much, much more

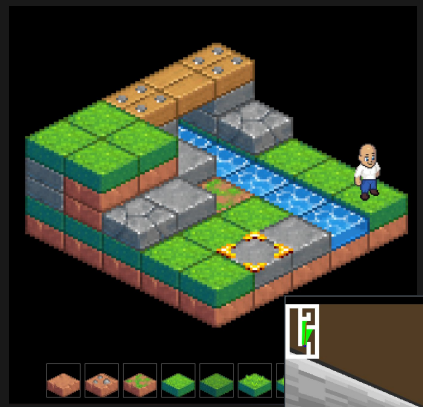
FLAME

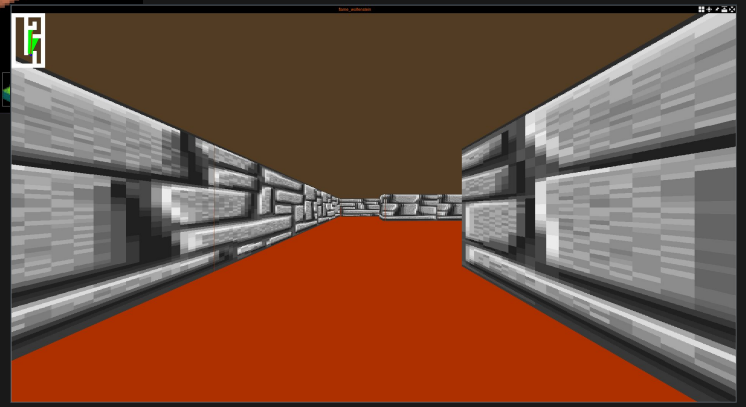


FLAME

Quick Recap - 2D Worlds - "2.5D" Techniques

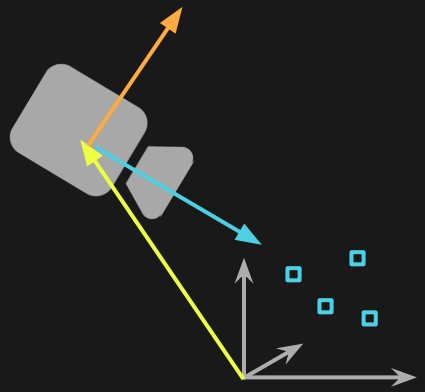
- Perspective
- Pre-rendered 3D sprites
- Parallax
- Isometric
- Sprite Stacking
- Raycasting





FLAME

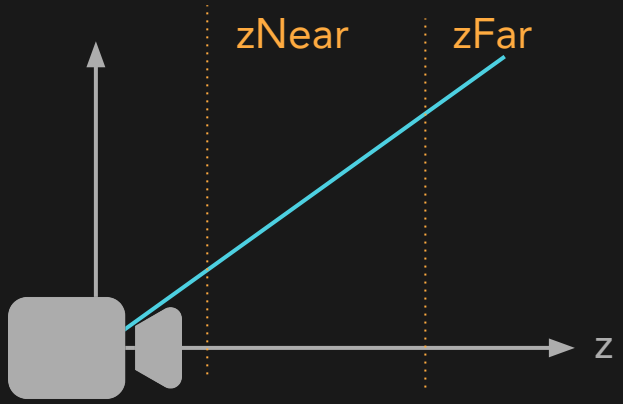
3D Projections - Camera

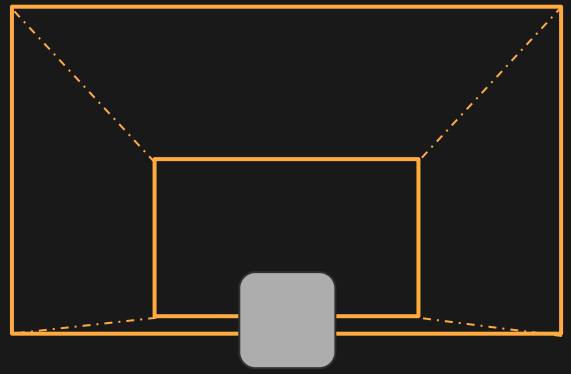


```
class Camera {
    Vector3 position;

    Vector3 direction;

    Vector3 up;

    // ...
}
```

FLAME

3D Projections - Camera





FLAME