

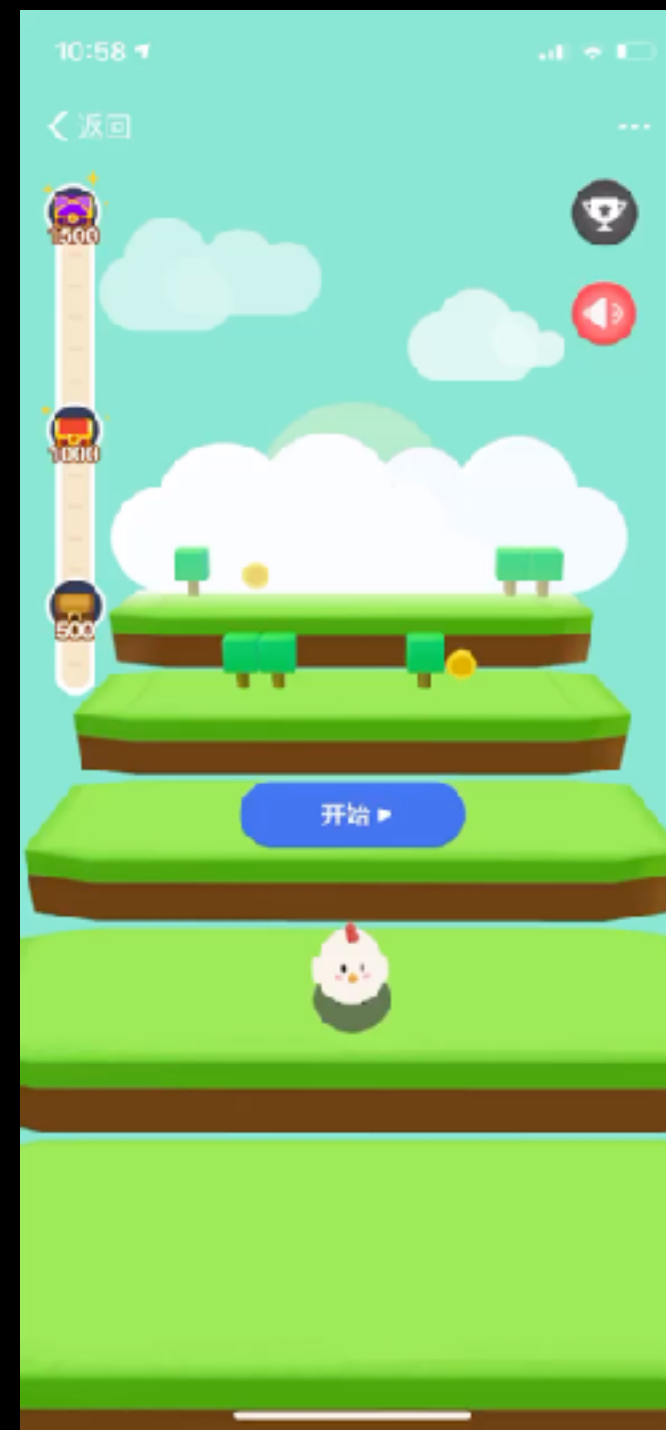
3rd
SEE Conf
蚂蚁金服体验科技大会

蚂蚁金服 Web 3D 技术 探索之路

烧鹅

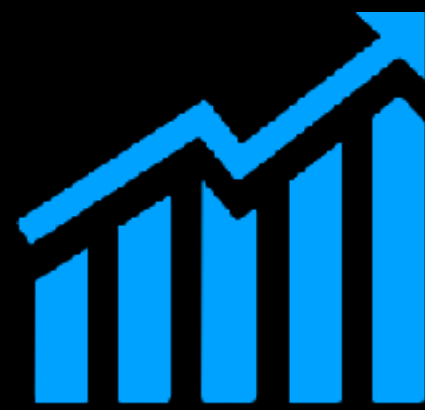
Oasis 3D 引擎负责人

Web 3D互动项目



业务背景

业务视角



提升业务的趣味性和商业转化数据



互联网产品
游戏化



技术视角



储备3D图形技术的开发能力

技术调研

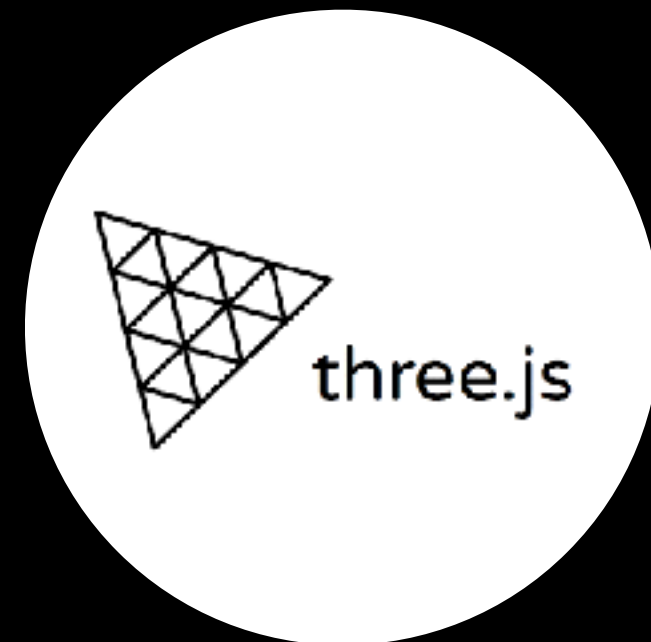
Web-based

- 非移动优先
- workflow 太弱

适合蚂蚁，技术深耕

Native-based

- 非 Web 优先
- 上手成本过高



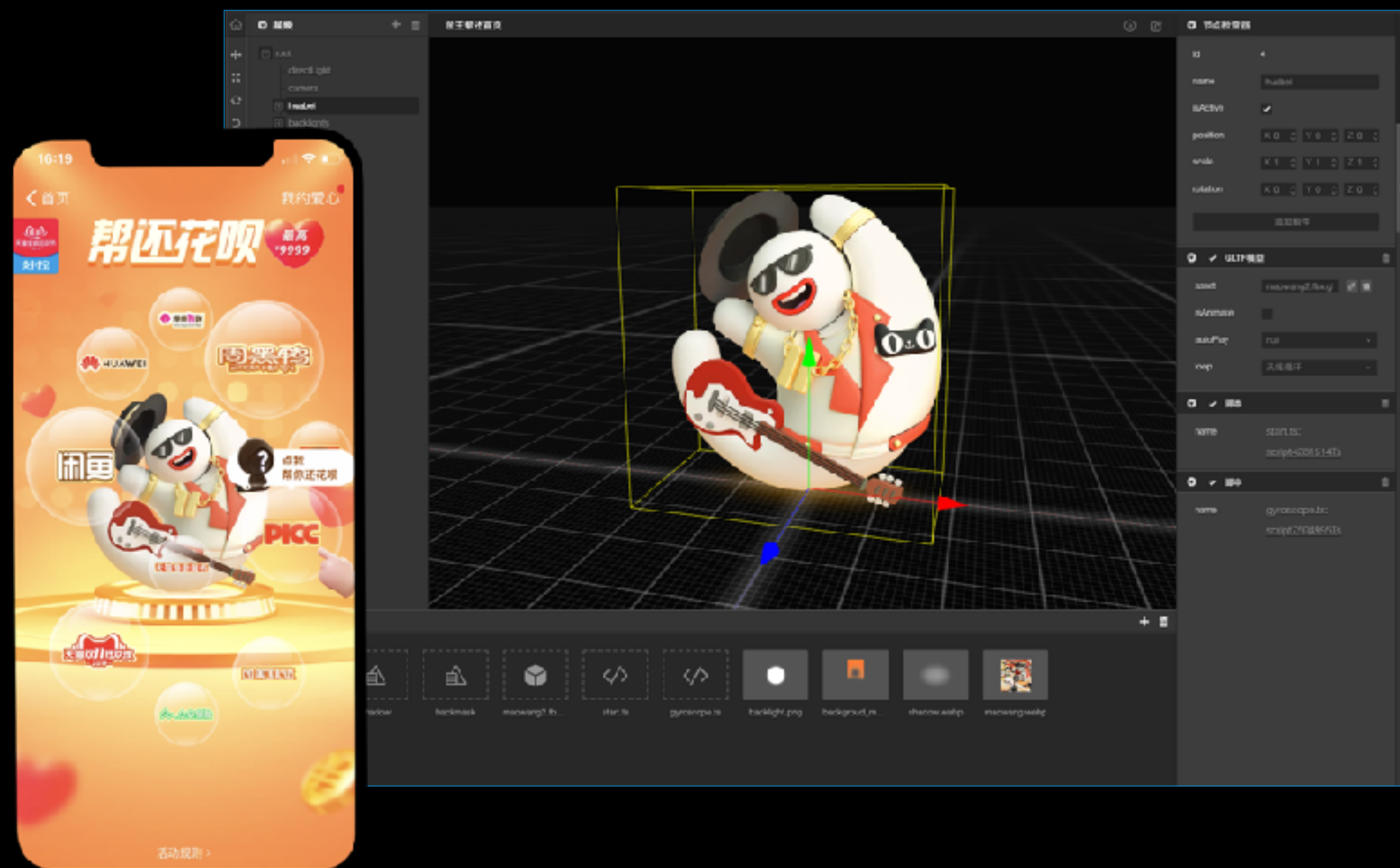
我们的方案

一个自研 3D 引擎

Oasis 3D

- 以 Web 为基础，面向前端工程师
- 移动优先，极致性能
- 适合蚂蚁金服的业务与技术发展

一个配套 workflow



目录

01 Oasis 3D 引擎

02 Oasis 3D workflow

03 未来展望

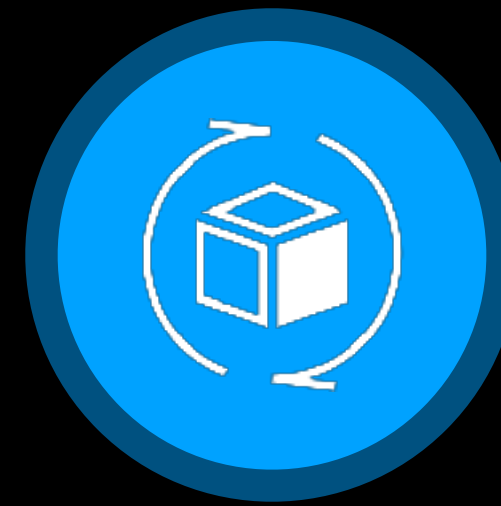
/01

Oasis 3D 引擎

Oasis 3D 引擎



架构



渲染



动画



架构

微内核架构

低耦合，高维护性，易组装，易扩展

ALINPM

能力扩展

Collider

Collision

Skybox

Animation

Camera

Light

Particle

Controls

设计模式

Engine

Scene

Node

Node-ability

Event-dispatcher

资产管理

Loader

Loader-gltf

Primitive

Geometry

Material

PBR

渲染管线

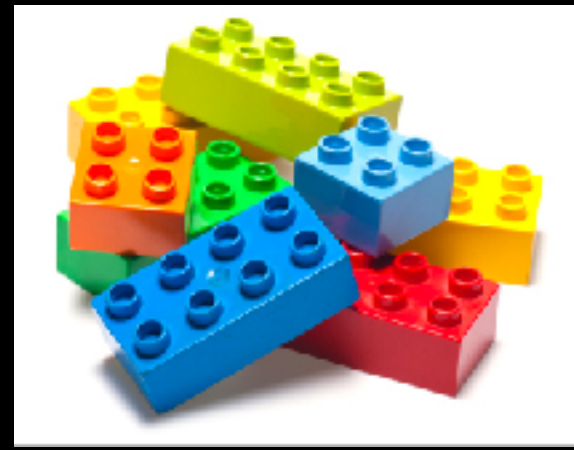
RHI

Render

Render-cull

Post-processing

微内核架构的优势



互动游戏引擎

工业产品渲染引擎

3D地理可视化引擎

针对不同业务场景组装特定引擎

渲染管线

WebGL1.0渲染

WebGL2.0渲染

WebGPU渲染

资产管理

gltf加载器

USD加载器

GeoJSON加载器

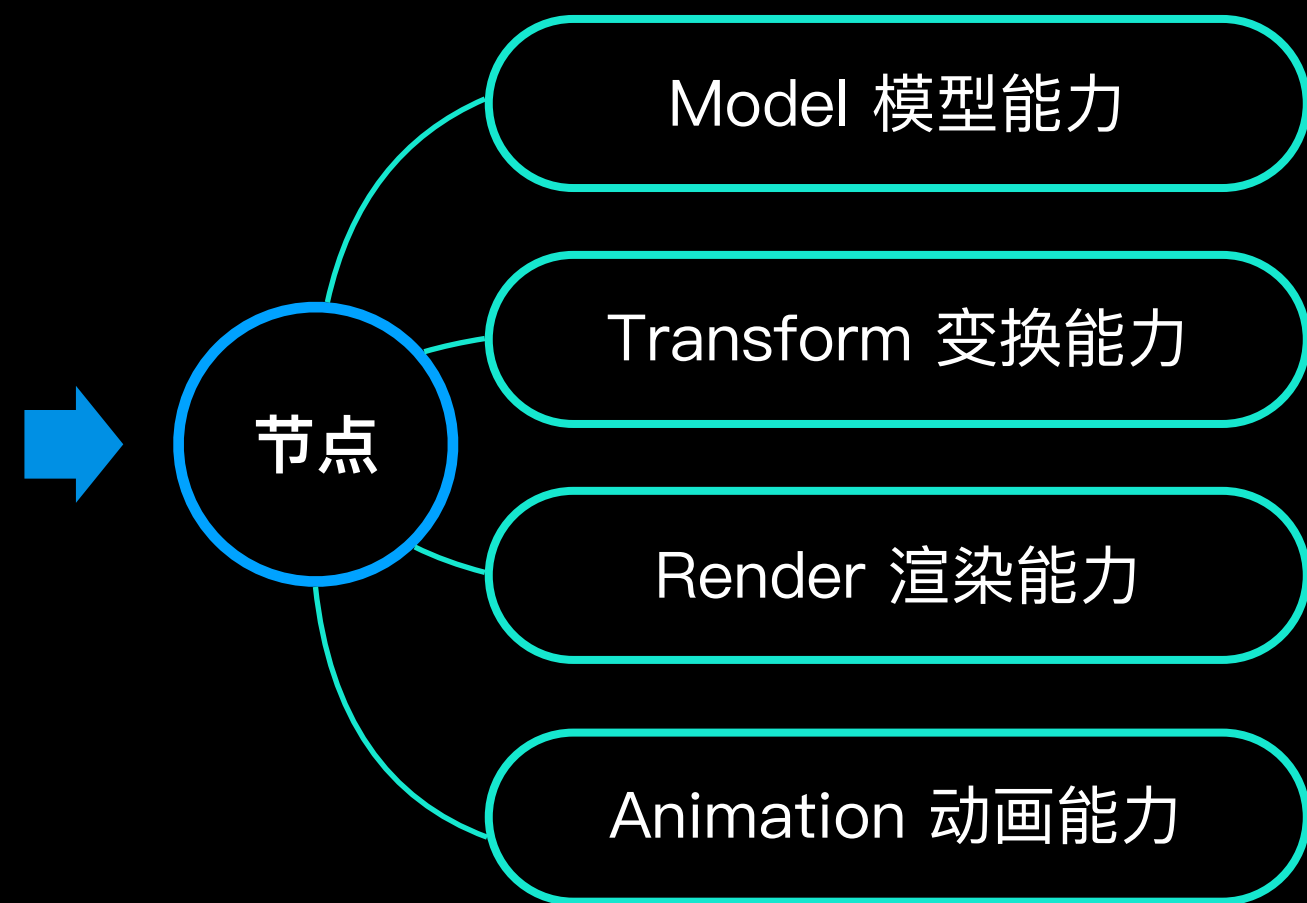
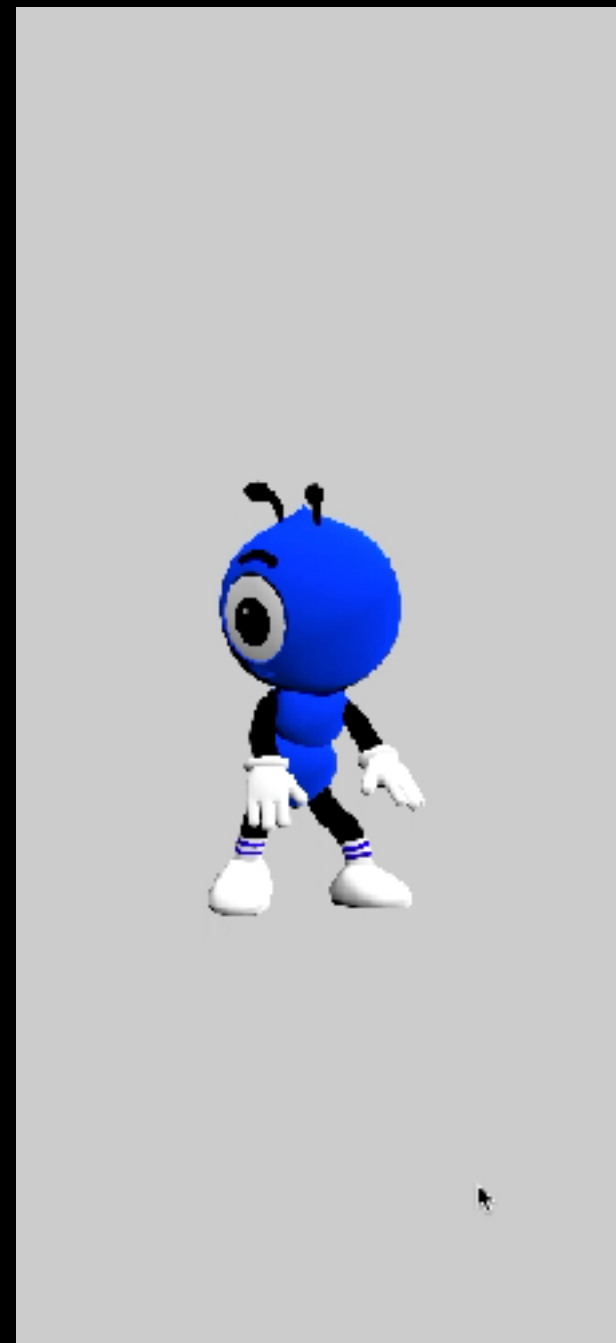
能力扩展

有限状态机

GPU Picker

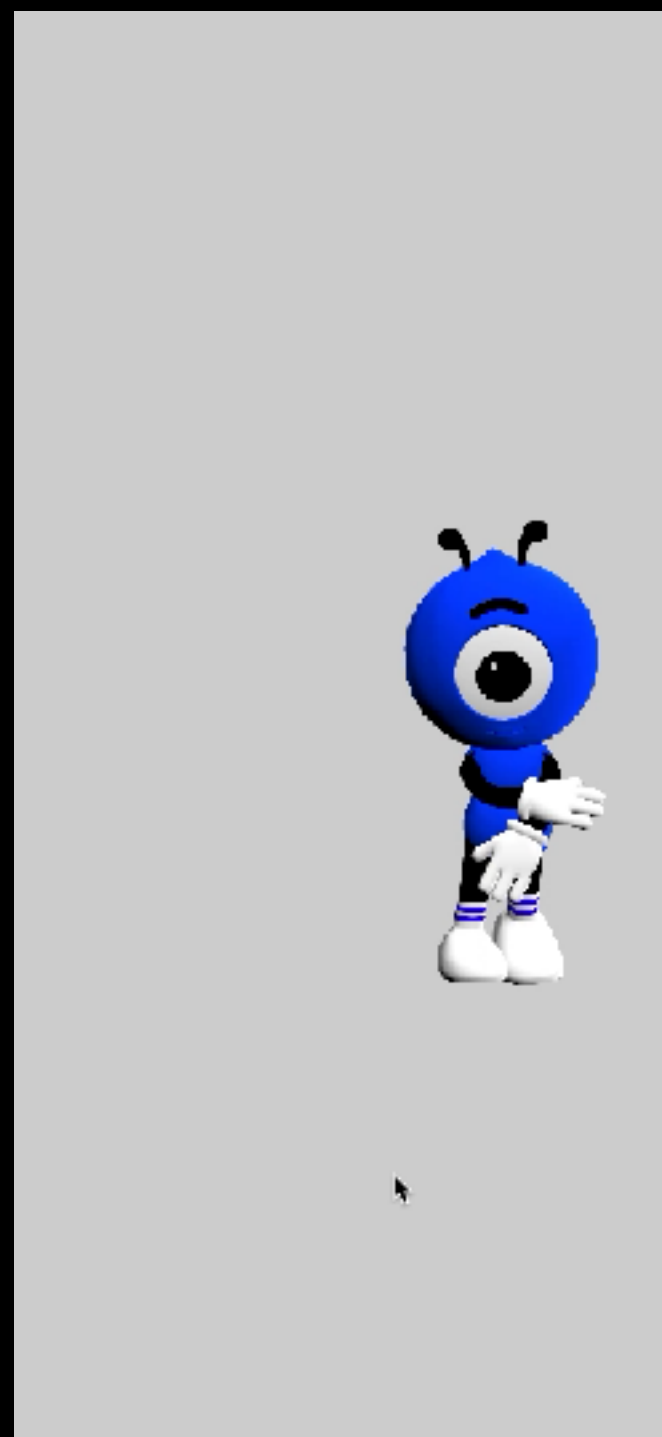
大气散射

内置能力扩展



```
const engine = new Engine();
const rootNode = engine.currentScene.root;
// 创建节点
const antNode = rootNode.createChild('ant_node');
// 创建能力
antNode.createAbility(GLTFAbility, {...});
antNode.createAbility(TransformAbility, {...});
antNode.createAbility(RenderAbility, {...});
antNode.createAbility(AnimationAbility, {...});
```


自定义能力扩展



```
// “麦克杰克逊”能力
class MichaelJacksonAbility extends NodeAbility
{
    constructor(node) {
        super(node);
        this._time = 0;
    }
    onUpdate(deltaTime) {
        this._time += deltaTime;
        let x = Math.sin(this._time * 0.001) * 4;
        this.node.position[0] = x;
    }
}

// 扩展“麦克杰克逊”能力
antNode.createAbility(MichaelJacksonAbility);
```

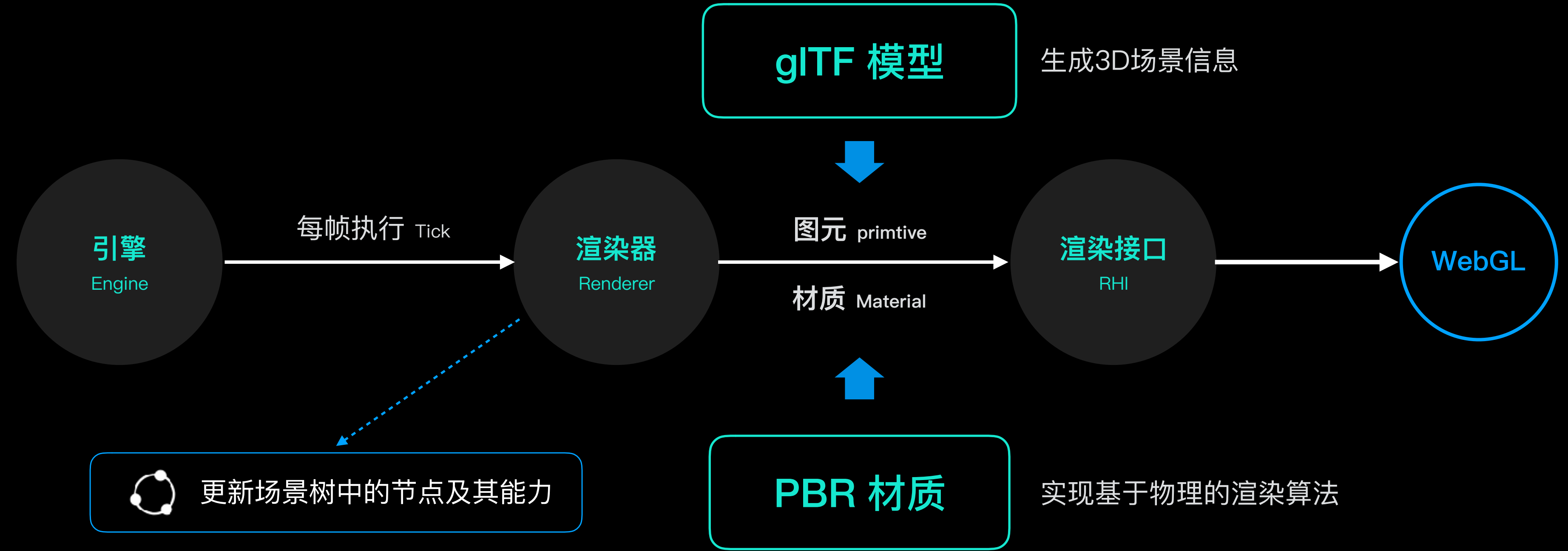
NodeAbility 是能力的基类

- onStart: 能力第一次更新时调用
- onUpdate: 每帧渲染之前被调用
- onDestroy: 能力销毁时调用



渲染

渲染管线设计



glTF: Web 3D 标准场景格式



Runtime 3D Asset Delivery

glTF™ (GL Transmission Format) is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by applications. glTF minimizes both the size of 3D assets, and the runtime processing needed to unpack and use those assets. glTF defines an extensible, common publishing format for 3D content tools and services that streamlines authoring workflows and enables interoperable use of content across the industry.



.gltf (JSON)

场景树、PBR 材质贴图、相机...

.bin

几何体: vertices and indices

动画: key-frames

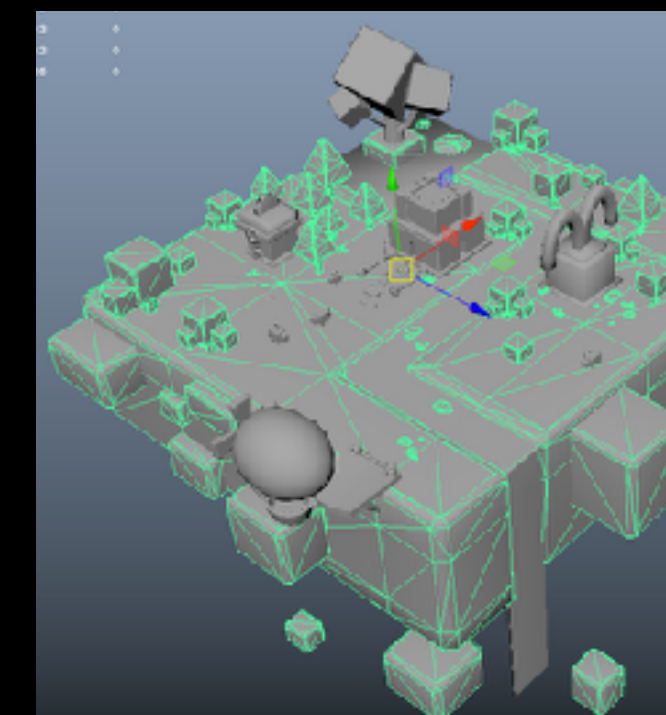
蒙皮: inverse-bind matrices

.png

.jpg

...

Textures



Geometry

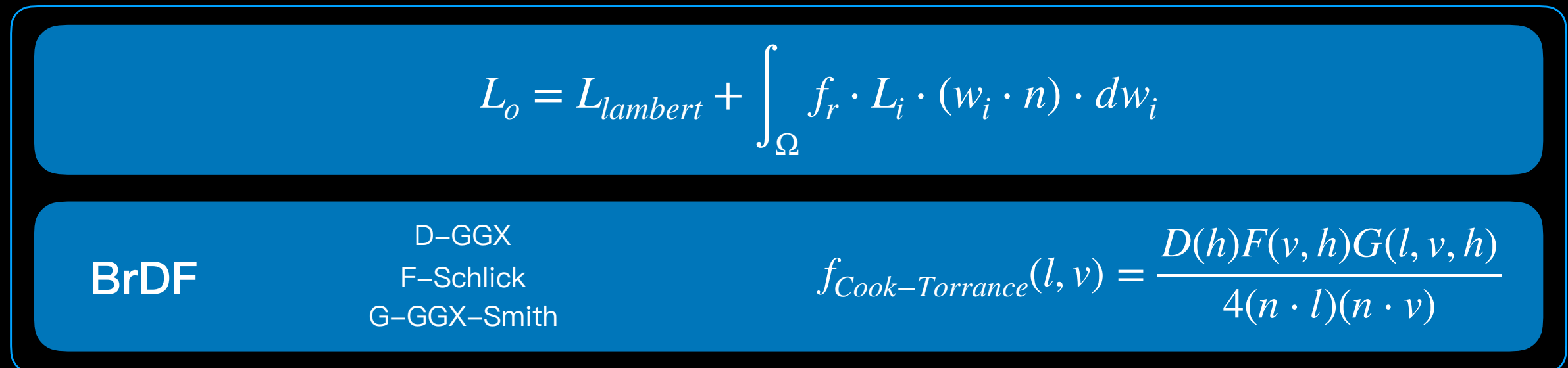
PBR: 基于物理的渲染



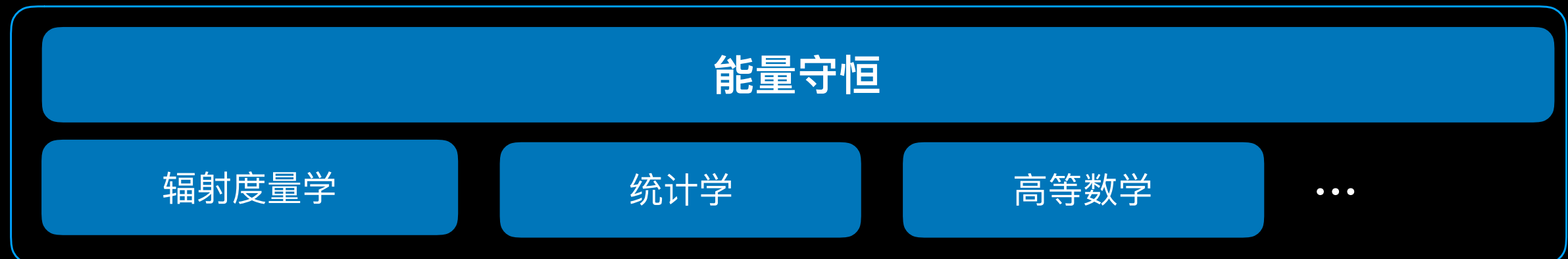
应用层



算法层



概念层



WebGL 1.0 渲染硬件接口

WebGL 从入门到放弃

Drawing a triangle with WebGL



You should see a pink canvas above with a cyan triangle in it. It's drawn with WebGL. This "hello world" takes around 25 lines, and involves lots of concepts like buffers and shaders.

```
const canvas = document.getElementById('triangleCanvas');
const gl = canvas.getContext('webgl');
gl.viewport(0, 0, canvas.width, canvas.height);

const vertShader = gl.createShader(gl.VERTEX_SHADER);
gl.shaderSource(vertShader, 'attribute vec3 c; void main(void){gl_Position=vec3(c, 1.0);}');
gl.compileShader(vertShader);

const fragShader = gl.createShader(gl.FRAGMENT_SHADER);
gl.shaderSource(fragShader, 'void main(void){gl_FragColor=vec4(0.1,1,1,1);}');
gl.compileShader(fragShader);

const prog = gl.createProgram();
gl.attachShader(prog, vertShader);
gl.attachShader(prog, fragShader);
gl.linkProgram(prog);
gl.useProgram(prog);

gl.clearColor(1, 0, 1, 1);
gl.clear(gl.COLOR_BUFFER_BIT);

const vertexBuf = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuf);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array([-0.5,0.5,0.0, -0.5,-0.5,0.0, 0.5,-0.5,0.0]), gl.STATIC_DRAW);

const coord = gl.getAttribLocation(prog, 'c');
gl.vertexAttribPointer(coord, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(coord);

gl.drawArrays(gl.TRIANGLES, 0, 3);
```

WebGL 1.0 API Quick Reference Card - Page 1

WebGL is a software interface for accessing graphic hardware from within a web browser. Based on OpenGL ES 2.0, WebGL allows a programmer to specify the objects and operations involved in producing high-quality graphical images, specifically color images of 3D objects.

- `gl` refers to sections in the WebGL 1.0 specification, available at www.khronos.org/webgl/
- Content marked in purple does not have an corresponding function in OpenGL ES 2.0 specification is available at www.khronos.org/registry/

WebGL function calls become identical to their OpenGL ES counterparts unless otherwise noted.

Interfaces

WebGL Context Attributes (3.1) The interface contains requested drawing surface attributes and is specified in the second parameter to `getContext`.

attributes: Default: that if you request drawing buffer with an application for the purpose of performing WebGL operations with a specific and is specified in the second parameter to `getContext`.

alpha: Default: that if you request drawing buffer with alpha buffer of at least 32bits.

depth: Default: that if you request drawing buffer of at least 16bit.

stencil: Default: that if you request drawing buffer with stencil buffer of at least 8bits.

antialias: Default: that if you request drawing buffer with antialiasing capabilities.

preserveDrawingBuffer: Default: that if you request drawing buffer which contains data with persistent data (persistent) after a tab is closed.

powerPreference: Default: that if you request context of the drawing surface to be power efficient or to be power efficient if available.

WebGL Objects (3.1)

This is the general interface for WebGL resources objects.

Texture Interface Objects

WebGLTexture (3.1) Object: Texture Object
WebGLTexture2D (3.1) Object: Texture Object
WebGLTexture3D (3.1) Object: Texture Object
WebGLImage (3.1) Object: Image Object
WebGLImageSubData (3.1) Object: Image Object
WebGLImageUnits (3.1) Object: Image Object
WebGLImageUnits (3.1) Object: Image Object
WebGLImageUnits (3.1) Object: Image Object

WebGL Buffer Objects (3.1)

WebGLBuffer (3.1) Object: Buffer Object
WebGLArrayBuffer (3.1) Object: Buffer Object
WebGLTypedArray (3.1) Object: Buffer Object
WebGLBuffer (3.1) Object: Buffer Object
WebGLBuffer (3.1) Object: Buffer Object
WebGLBuffer (3.1) Object: Buffer Object

WebGL Shader Objects (3.1)

WebGLShader (3.1) Object: Shader Object
WebGLProgram (3.1) Object: Shader Object
WebGLShaderPrecisionFormat (3.1) Object: Shader Object
WebGLShaderPrecisionFormat (3.1) Object: Shader Object
WebGLShaderPrecisionFormat (3.1) Object: Shader Object

WebGL Render Context (3.1)

WebGLRenderingContext (3.1) Object: Render Context Object
WebGLRenderingContext (3.1) Object: Render Context Object
WebGLRenderingContext (3.1) Object: Render Context Object

WebGL Error Codes (3.1)

WebGLError (3.1) Object: Error Object
WebGLError (3.1) Object: Error Object
WebGLError (3.1) Object: Error Object

WebGL Extensions (3.1)

WebGLExtension (3.1) Object: Extension Object
WebGLExtension (3.1) Object: Extension Object
WebGLExtension (3.1) Object: Extension Object

WebGL 1.0 API Quick Reference Card - Page 2

Programs and Shaders (3.1)

gl.createShader(shaderType) (3.1.1) Creates a new shader object of the specified type.

gl.shaderSource(shader, source) (3.1.2) Specifies the source code for the specified shader object.

gl.compileShader(shader) (3.1.3) Compiles the source code for the specified shader object.

gl.getShaderPrecisionFormat(shaderType) (3.1.4) Returns the precision format for the specified shader type.

gl.createProgram() (3.1.5) Creates a new program object.

gl.attachShader(program, shader) (3.1.6) Attaches the specified shader object to the specified program object.

gl.linkProgram(program) (3.1.7) Links the specified program object.

gl.validateProgram(program) (3.1.8) Validates the specified program object.

gl.useProgram(program) (3.1.9) Enables the specified program object.

gl.deleteProgram(program) (3.1.10) Deletes the specified program object.

gl.isProgram(program) (3.1.11) Returns true if the specified program object is a program object.

Texture Objects (3.1)

gl.texImage2D(target, level, internalFormat, width, height, border, format, type, srcData) (3.1.1) Generates a 2D texture from a 2D image.

gl.texImage3D(target, level, internalFormat, width, height, depth, border, format, type, srcData) (3.1.2) Generates a 3D texture from a 3D image.

gl.texImage2D(target, level, internalFormat, width, height, border, format, type, srcData) (3.1.3) Generates a 2D texture from a 2D image.

gl.texImage3D(target, level, internalFormat, width, height, depth, border, format, type, srcData) (3.1.4) Generates a 3D texture from a 3D image.

gl.texImage2D(target, level, internalFormat, width, height, border, format, type, srcData) (3.1.5) Generates a 2D texture from a 2D image.

gl.texImage3D(target, level, internalFormat, width, height, depth, border, format, type, srcData) (3.1.6) Generates a 3D texture from a 3D image.

gl.texImage2D(target, level, internalFormat, width, height, border, format, type, srcData) (3.1.7) Generates a 2D texture from a 2D image.

gl.texImage3D(target, level, internalFormat, width, height, depth, border, format, type, srcData) (3.1.8) Generates a 3D texture from a 3D image.

gl.texImage2D(target, level, internalFormat, width, height, border, format, type, srcData) (3.1.9) Generates a 2D texture from a 2D image.

gl.texImage3D(target, level, internalFormat, width, height, depth, border, format, type, srcData) (3.1.10) Generates a 3D texture from a 3D image.

Special Functions (3.1)

gl.getActiveAttrib(program, index) (3.1.1) Returns the name of the active attribute at the specified index.

gl.getActiveUniform(program, index) (3.1.2) Returns the name of the active uniform at the specified index.

gl.getAttribLocation(program, name) (3.1.3) Returns the index of the attribute with the specified name.

gl.getUniformLocation(program, name) (3.1.4) Returns the location of the uniform with the specified name.

gl.isAttribEnabled(program, index) (3.1.5) Returns true if the attribute at the specified index is enabled.

gl.isUniformEnabled(program, index) (3.1.6) Returns true if the uniform at the specified index is enabled.

gl.isTextureEnabled(target, level) (3.1.7) Returns true if the texture at the specified target and level is enabled.

gl.isTextureUnitEnabled(index) (3.1.8) Returns true if the texture unit at the specified index is enabled.

gl.isVertexArrayEnabled(index) (3.1.9) Returns true if the vertex array at the specified index is enabled.

gl.isVertexArrayEnabled(index) (3.1.10) Returns true if the vertex array at the specified index is enabled.

Renderbuffer Objects (3.1)

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.1) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.2) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.3) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.4) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.5) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.6) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.7) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.8) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.9) Generates a renderbuffer object.

gl.renderbufferStorage(target, internalFormat, width, height, depth) (3.1.10) Generates a renderbuffer object.

Writing to the Draw Buffer (3.1)

gl.drawArrays(mode, first, count) (3.1.1) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.2) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.3) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.4) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.5) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.6) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.7) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.8) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.9) Draws a set of vertices from an array to produce a set of primitives.

gl.drawArrays(mode, first, count) (3.1.10) Draws a set of vertices from an array to produce a set of primitives.

Framebuffer Objects (3.1)

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.1) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.2) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.3) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.4) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.5) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.6) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.7) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.8) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.9) Attaches a renderbuffer object to a framebuffer object.

gl.framebufferRenderbuffer(target, attachment, internalFormat, width, height) (3.1.10) Attaches a renderbuffer object to a framebuffer object.

View and Cull (3.1)

gl.viewport(x, y, width, height) (3.1.1) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.2) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.3) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.4) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.5) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.6) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.7) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.8) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.9) Specifies the viewport.

gl.viewport(x, y, width, height) (3.1.10) Specifies the viewport.

Rasterization (3.1)

gl.cullFace(mode) (3.1.1) Specifies the culling mode.

gl.cullFace(mode) (3.1.2) Specifies the culling mode.

gl.cullFace(mode) (3.1.3) Specifies the culling mode.

gl.cullFace(mode) (3.1.4) Specifies the culling mode.

gl.cullFace(mode) (3.1.5) Specifies the culling mode.

gl.cullFace(mode) (3.1.6) Specifies the culling mode.

gl.cullFace(mode) (3.1.7) Specifies the culling mode.

gl.cullFace(mode) (3.1.8) Specifies the culling mode.

gl.cullFace(mode) (3.1.9) Specifies the culling mode.

gl.cullFace(mode) (3.1.10) Specifies the culling mode.

Detected context lost events (3.1)

gl.getExtension("ANGLE_instanced_arrays") (3.1.1) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.2) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.3) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.4) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.5) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.6) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.7) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.8) Returns the ANGLE_instanced_arrays extension.

gl.getExtension("ANGLE_instanced_arrays") (3.1.9) Returns the ANGLE_instanced_arrays extension.

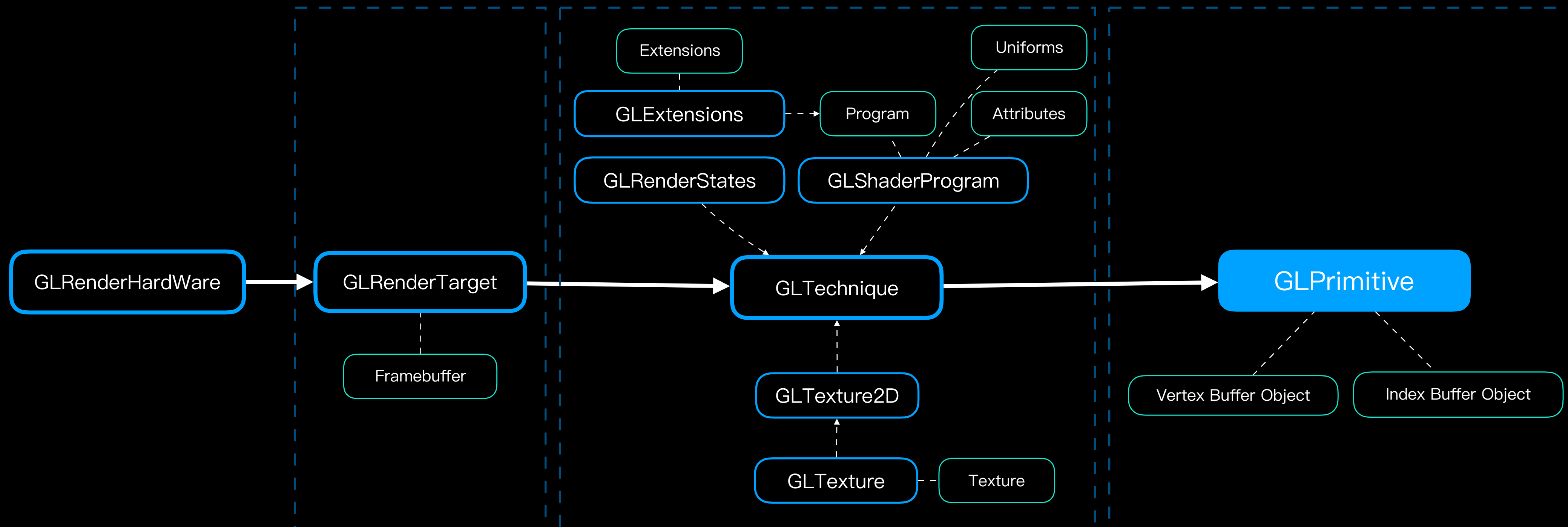
gl.getExtension("ANGLE_instanced_arrays") (3.1.10) Returns the ANGLE_instanced_arrays extension.

WebGL1.0 渲染硬件接口

找好画布

调好画笔

下笔画





动画

动画分类



花小呗/帽子

 骨骼动画

C4D / Maya

红包

 粒子动画

粒子系统能力

魔法棒、光效

 Shader动画

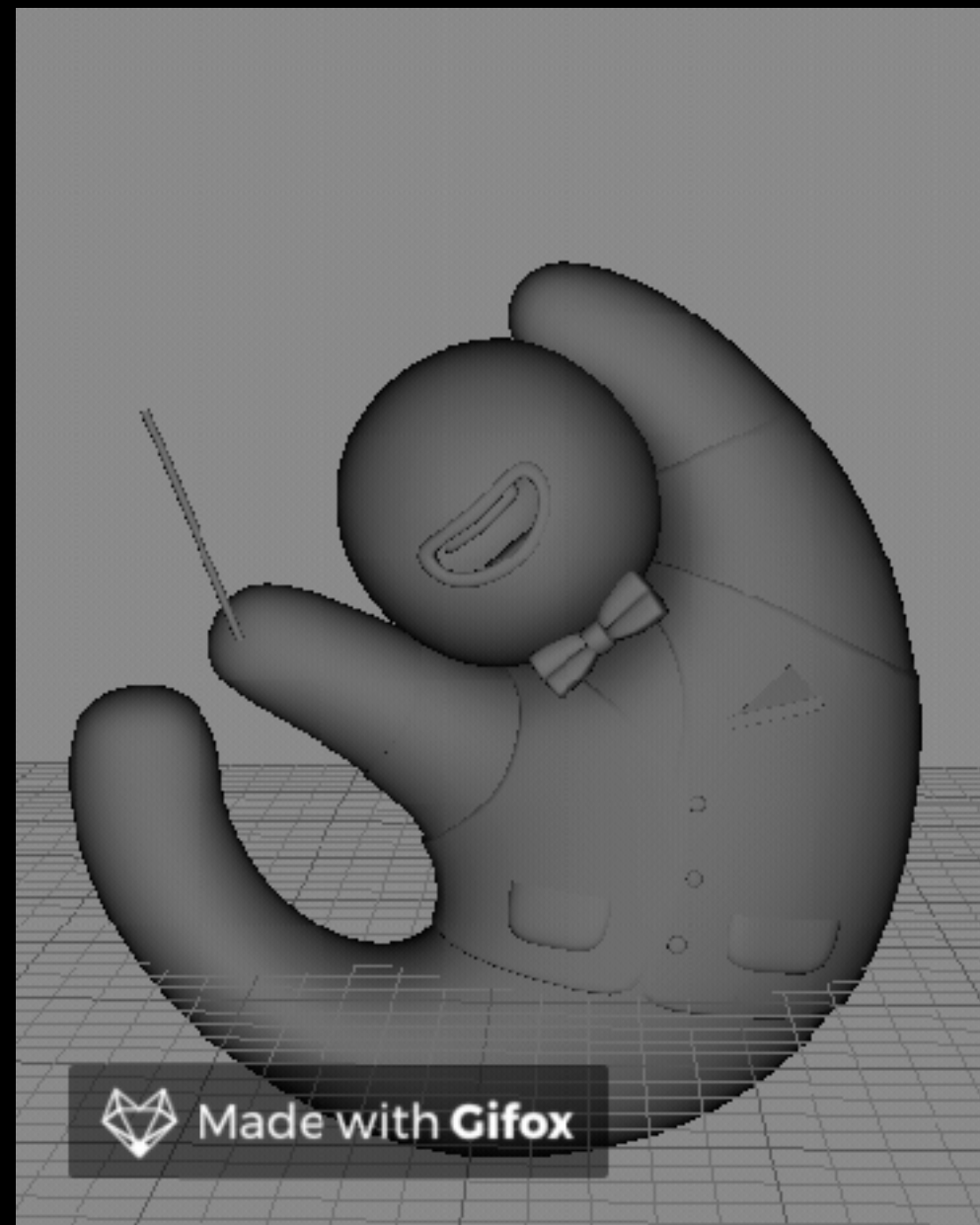
自定义 Shader 能力

烟雾

 帧动画

帧动画能力

骨骼动画



Oasis 3D
能力

MeshRenderAbility
绘制骨骼上的蒙皮纹理

AnimationAbility
控制骨骼动画的播放



骨架
Skeleton

4x4 Matrix

骨骼
Bone

影响

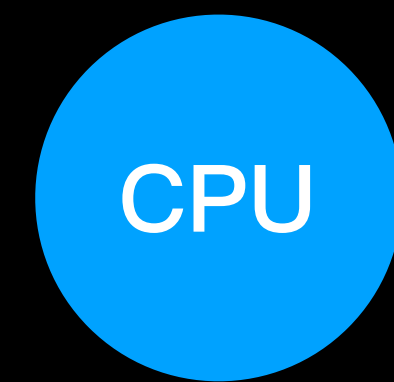
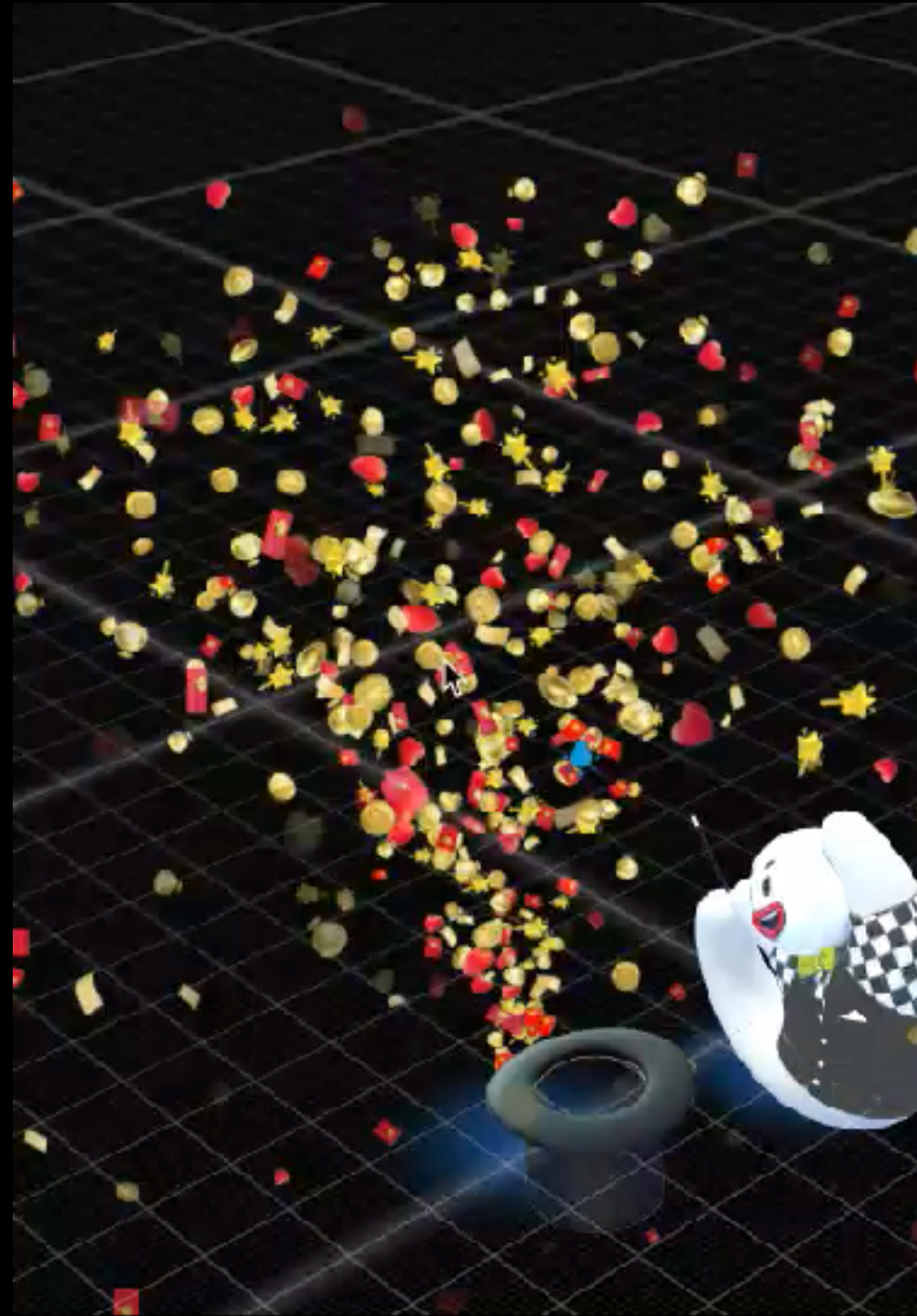
注意移动端骨骼数限制!

蒙皮
Skin

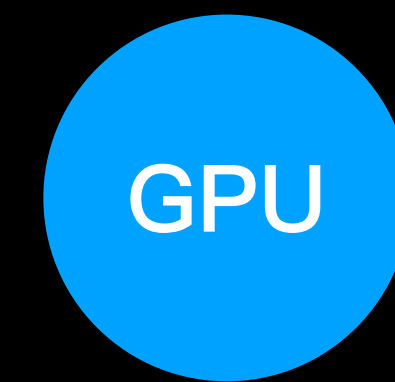
权重系数

顶点
Vertex

粒子动画



负责更新时间



负责更新粒子的状态

Position Rotation Scale Alpha Texture ...

匀变速运动

$$x = v \cdot t + \frac{1}{2} \cdot a \cdot t^2$$

Oasis 3D 引擎总结

- 微内核架构，组件式扩展
- glTF2.0 及配套 PBR 材质支持
- 丰富多样的动画系统

/02

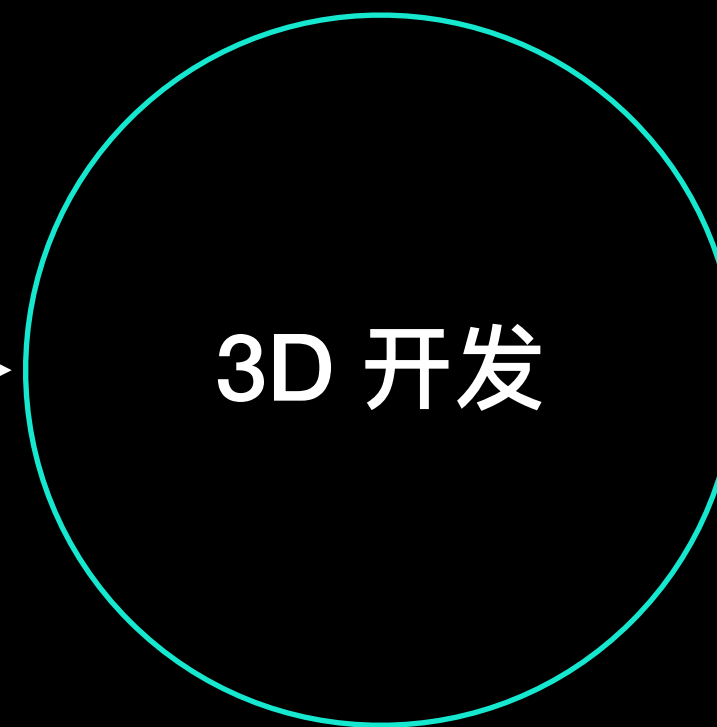
Oasis 3D workflow

2020年，仅有引擎够了吗？



流程繁琐 **3**天

美术资产进引擎困难重重



上手成本高 **5**天

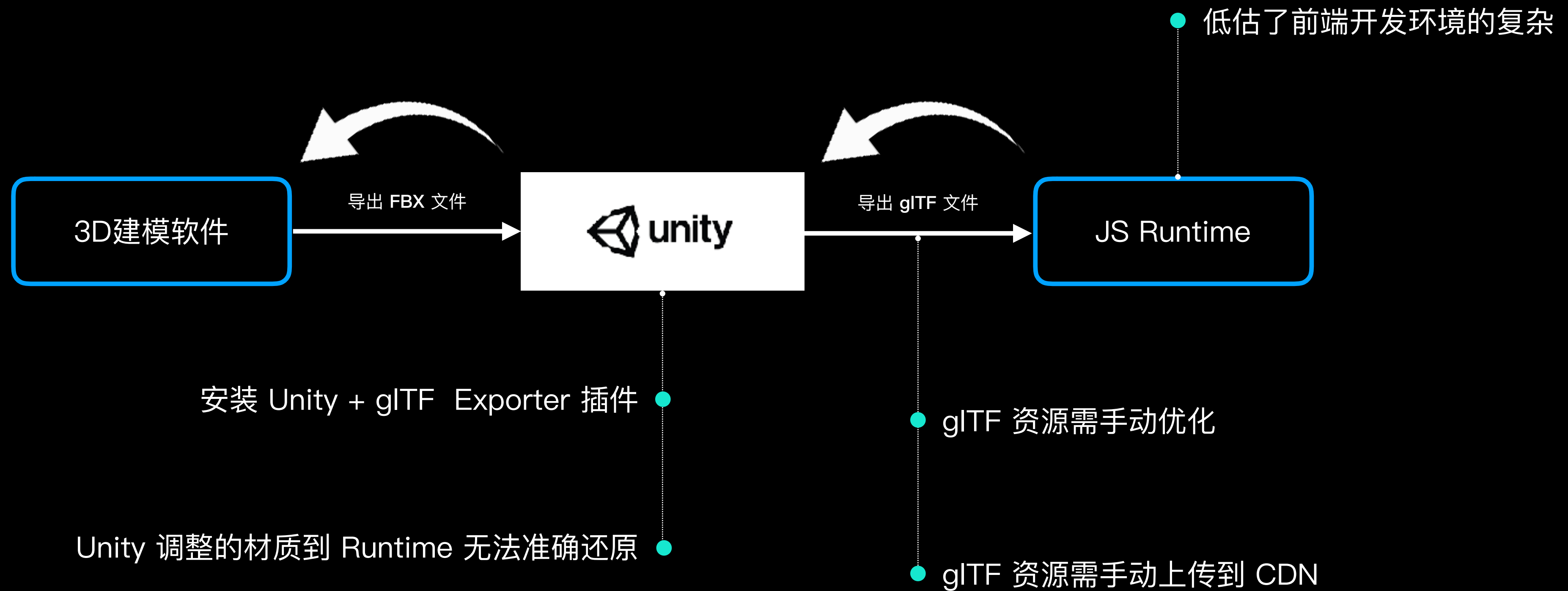
编写大量样板式代码



反复联调 **2**天

与前端代码缺少好的解耦规范

workflow 1.0



workflow 2.0

在线编辑器开箱即用

美术资产无缝对接

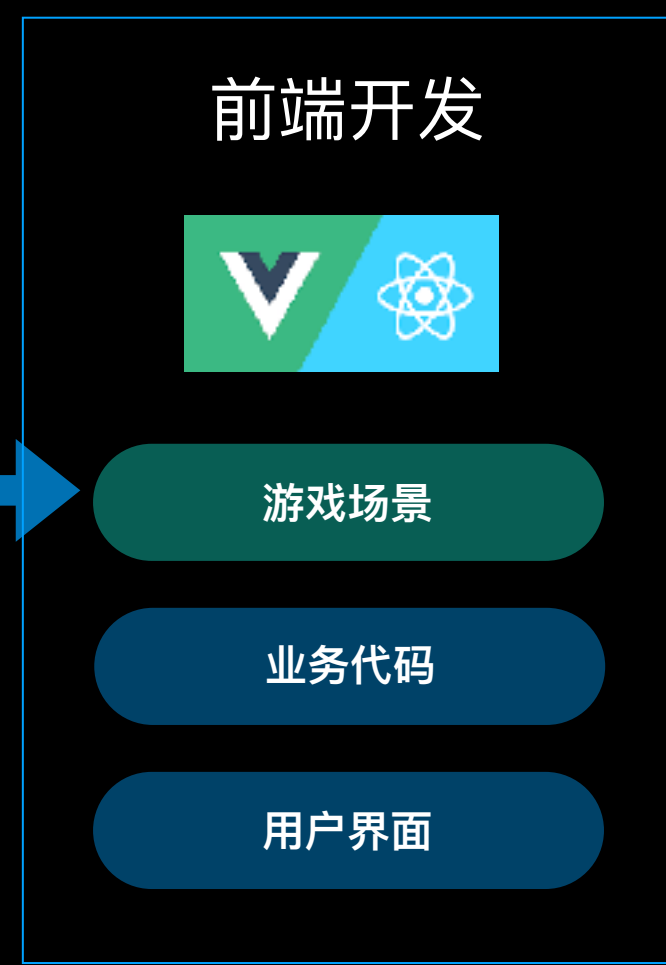


导出
FBX



导出
场景

面向前端工程设计



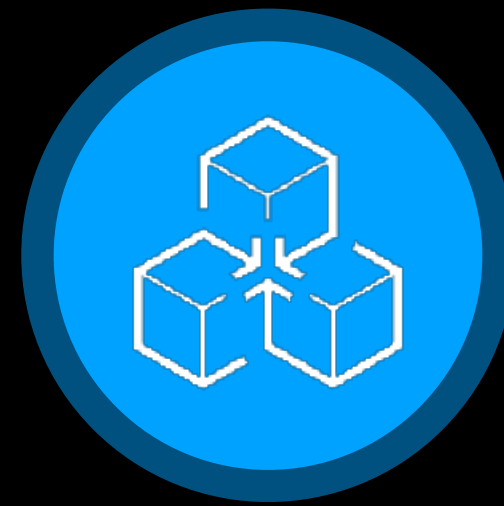
扫码
预览



Oasis 3D workflow



资产



编排



协作



资产

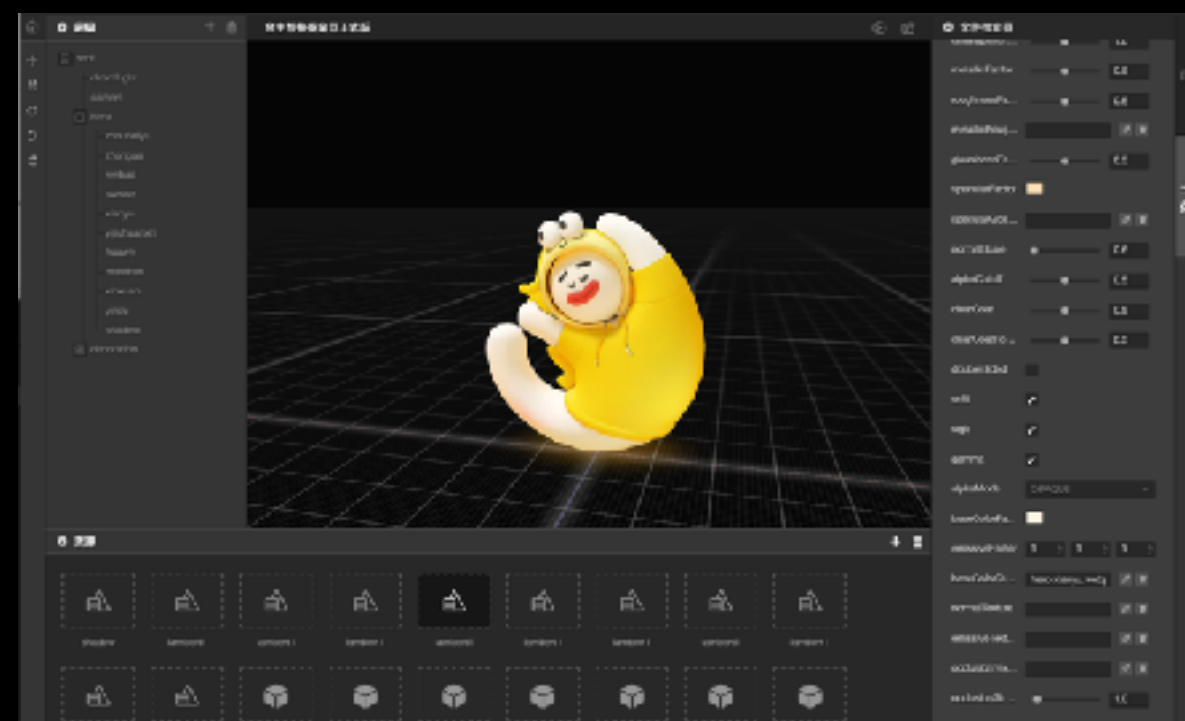
资产云端标准化

3D 美术资产如何快速进引擎?



资产沉淀与消费

资产调节管线



发布



蚂蚁3D资产市场

建设中

模型库

动作库

灯光库

材质库

引用



编辑器“库”面板

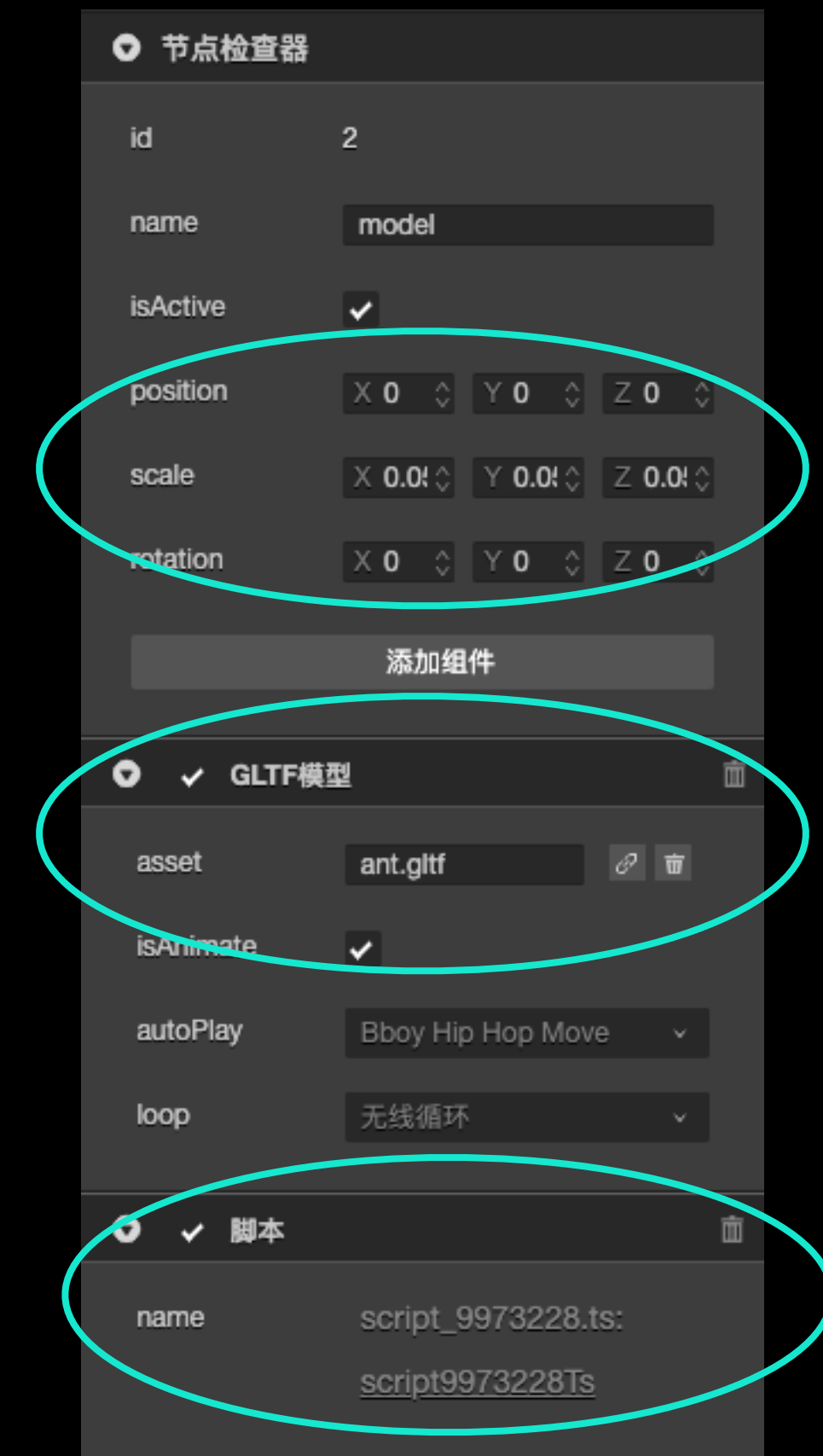
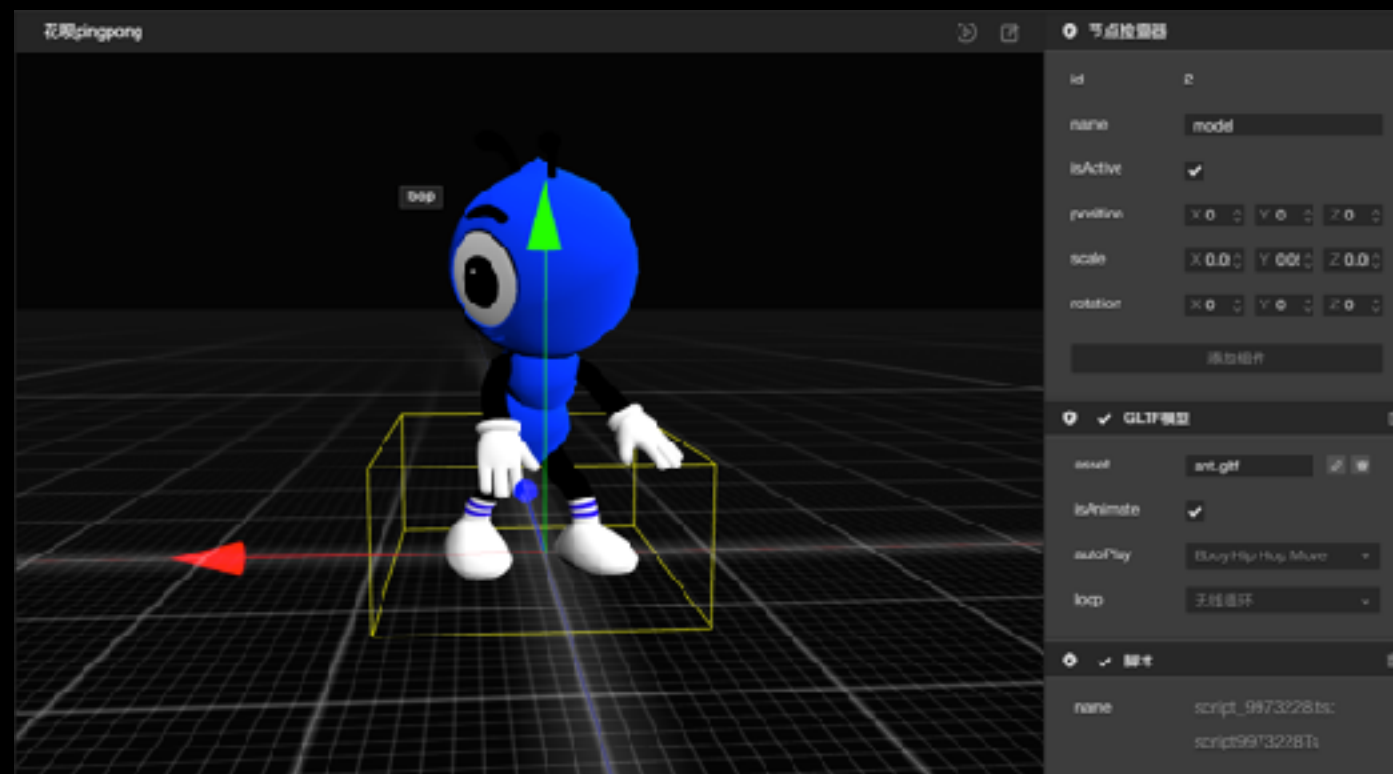
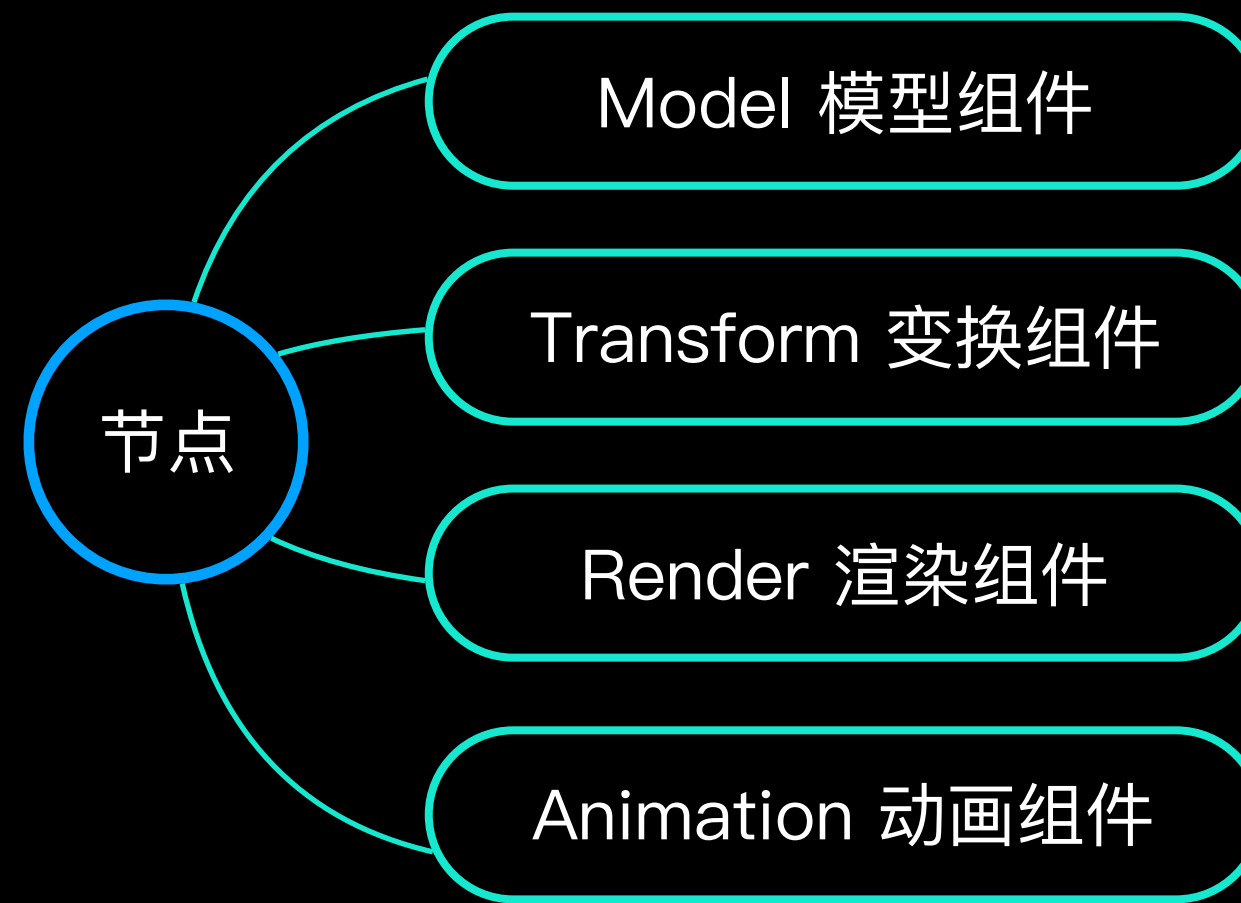




编排

能力编排

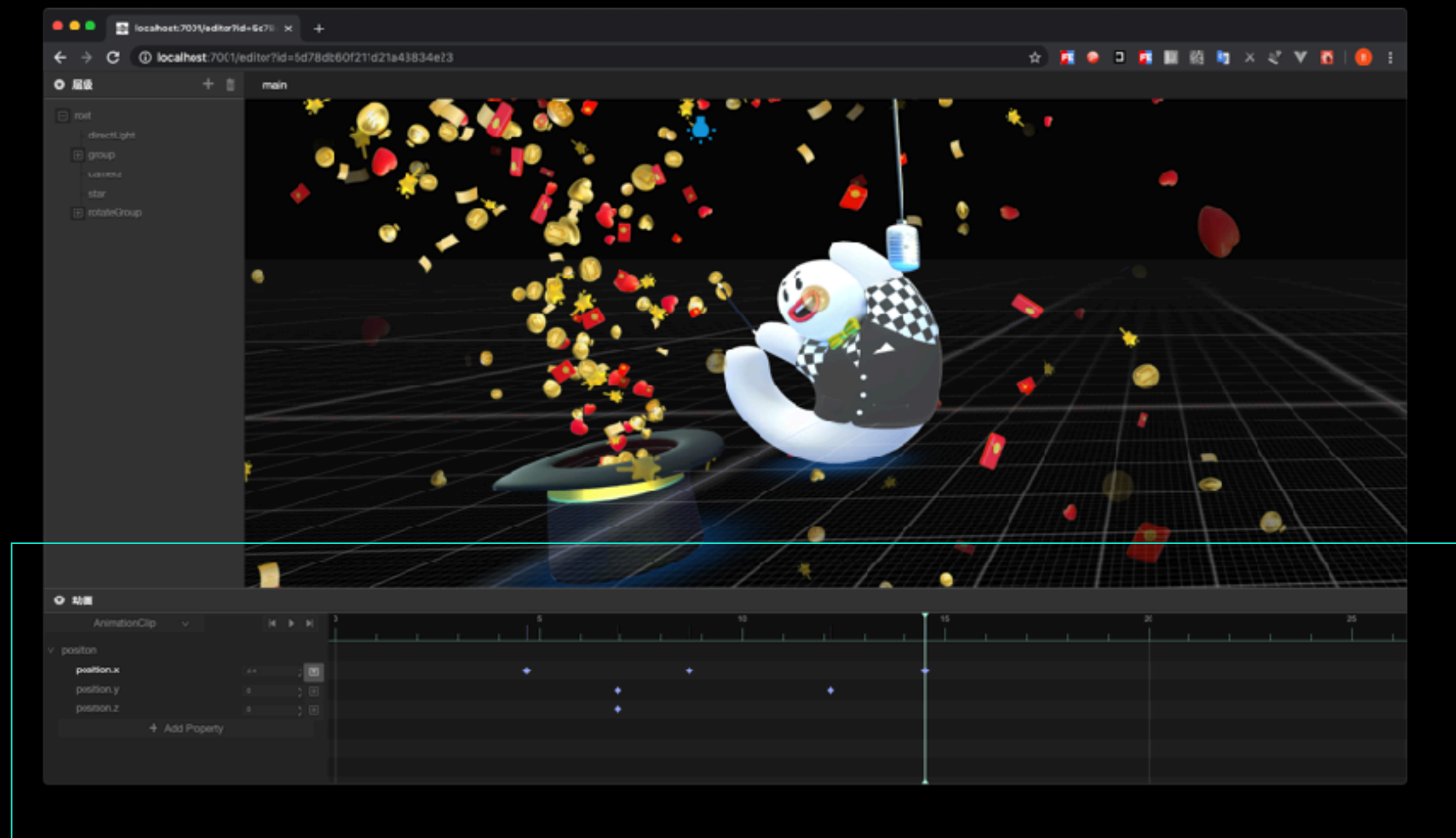
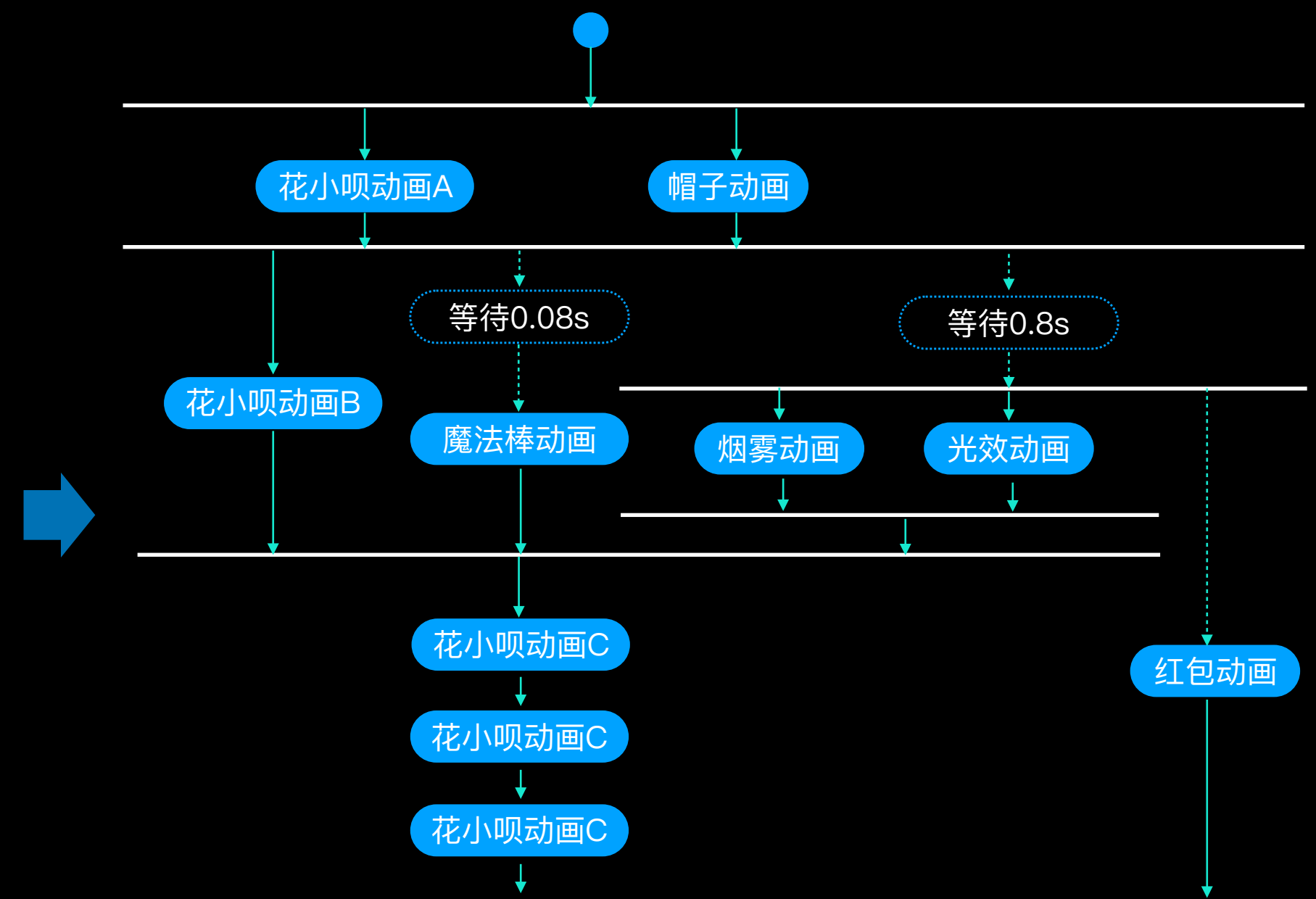
不用写样板式代码啦!



动画编排

- 还原难：过程式地编写动画代码
- 维护难：设计师动效改动频繁

Oasis Editor 时间轴面板





协作

设计协作

设计师

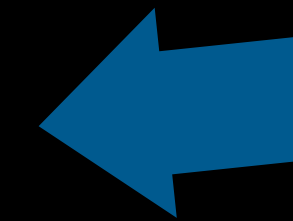
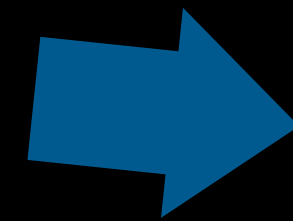
我觉得花小呗高光部分太亮了

哦，发你个地址，你自己调吧

工程师



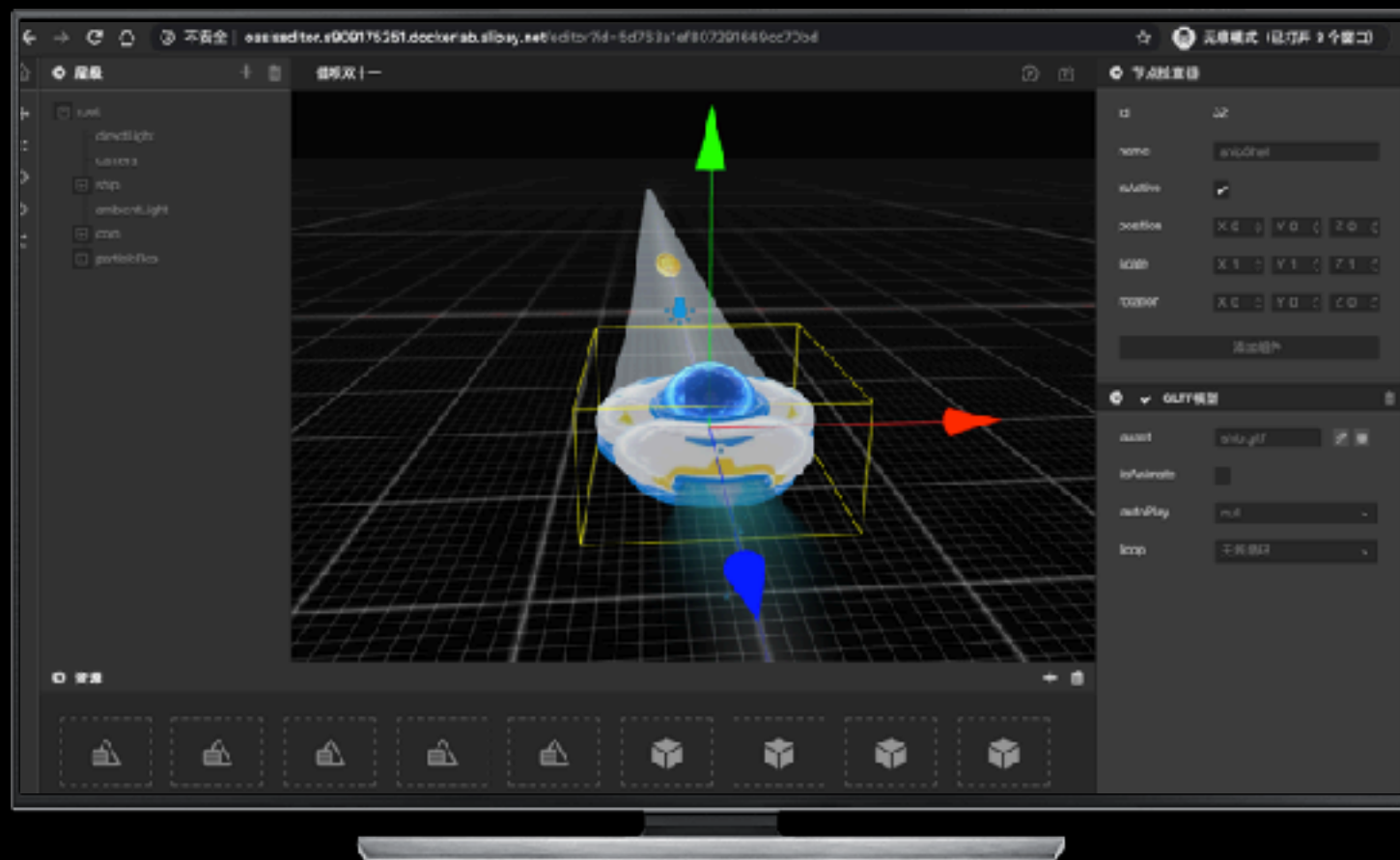
北京



杭州

前端协作

Oasis Editor

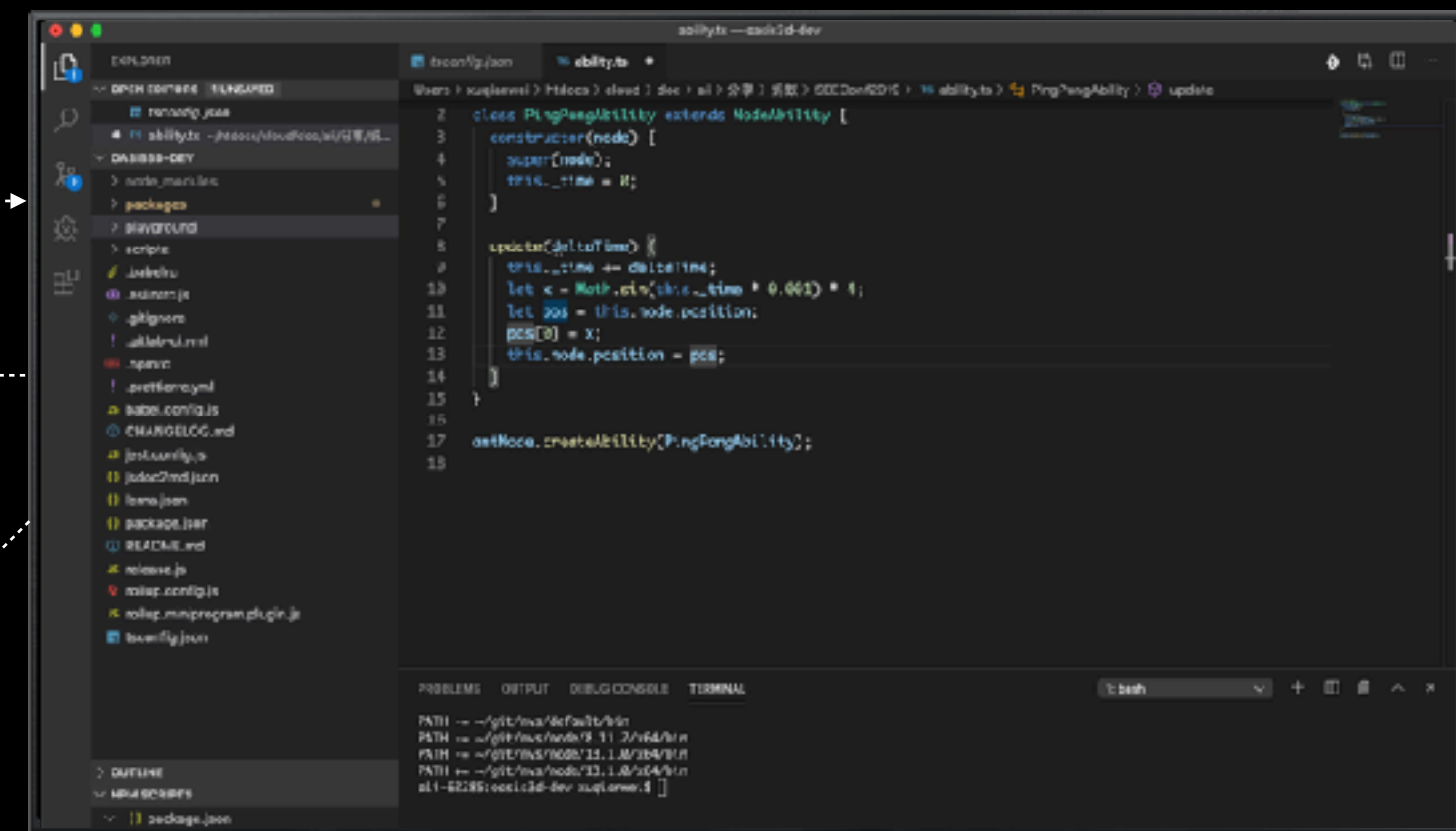


同步场景

Web Socket

同步代码

Webpack Server



Oasis 3D workflow总结

- 资产云端标准化，打通资产市场
- 能力编排和动画编排可视化
- 与设计和前端开发顺畅协作

/03
未来展望

引擎技术展望

布局三大业务引擎

互动游戏引擎

工业产品引擎

3D数据可视化引擎

高级渲染

WebGL2.0

WebGPU

风格化渲染

正向/延迟渲染

体渲染

全局光照

高性能

Web Worker

WebAssembly

GPGPU

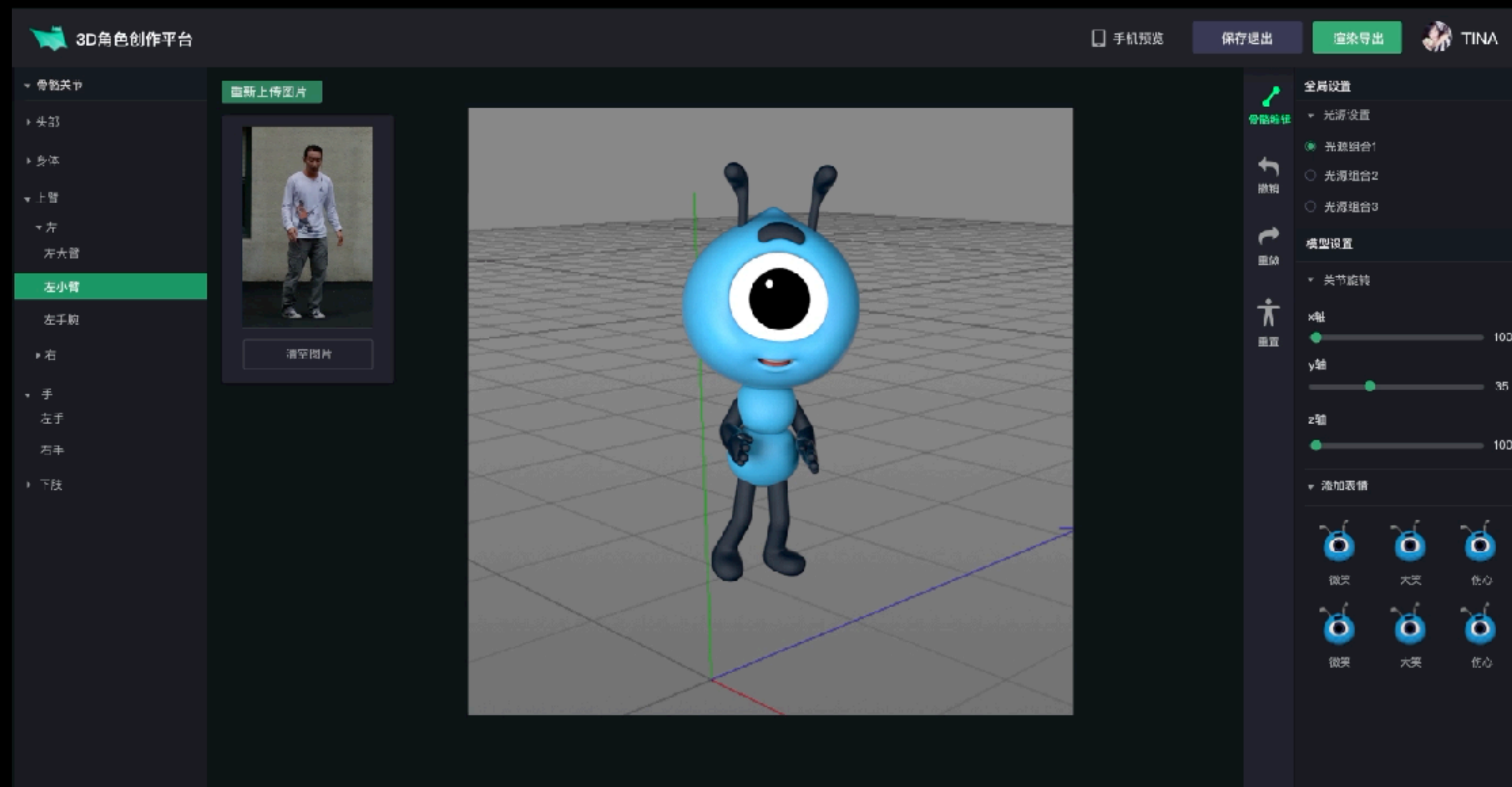
Simplify

Merge

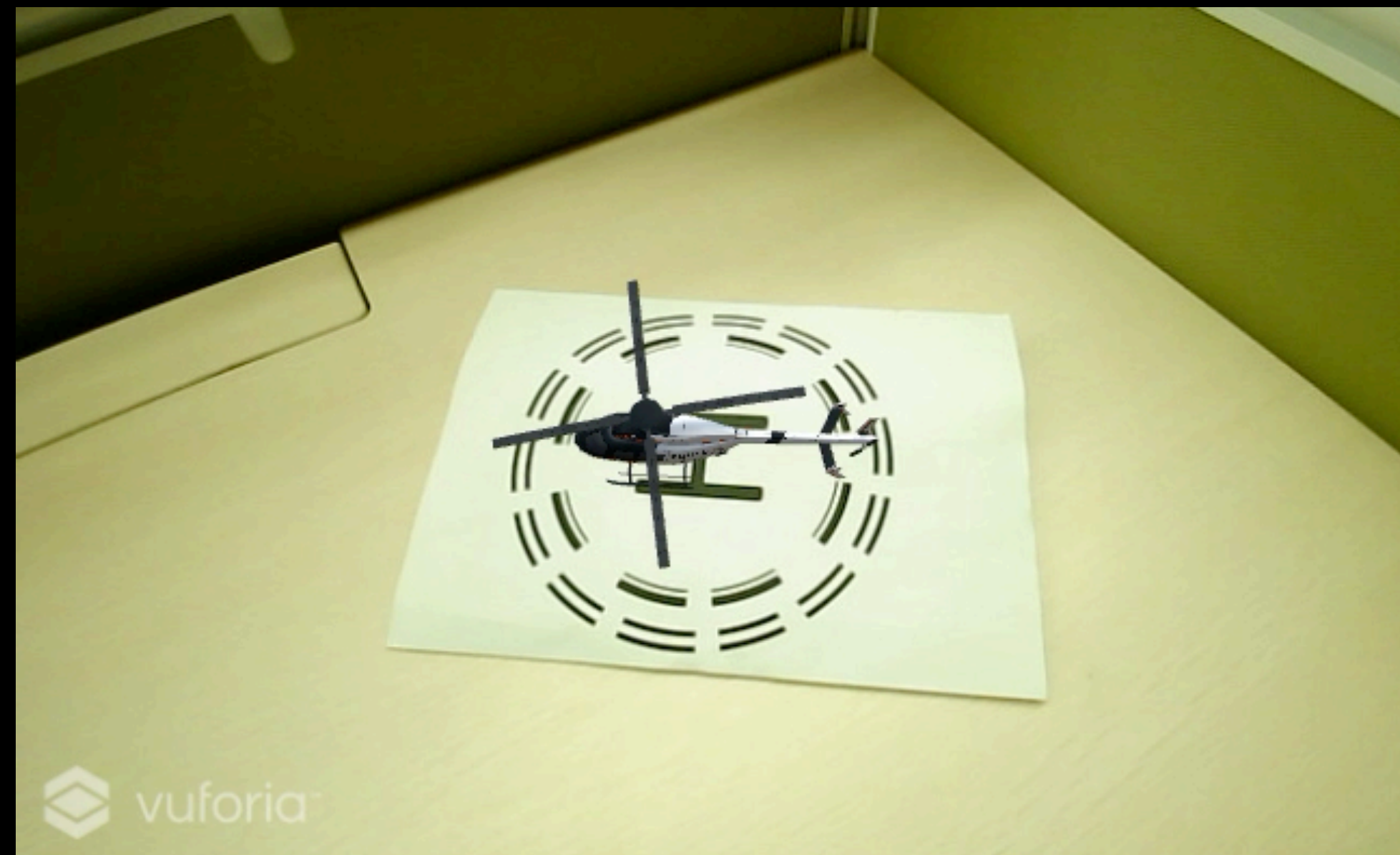
Culling

LOD

视觉算法生成动作素材



手势计算辅助3D交互



打通AR眼镜生态



长远目标

半年后

建立蚂蚁互动3D开发标准 workflow，落地工业产品渲染场景；对外开放引擎。

一年后

成立北京研发中心；正式开源引擎，开放 workflow。

两年后

建立一个多元化的综合团队，具备3D建模、游戏策划、VFX特效制作、创意编程等能力。

三年后

跻身世界一流3D引擎，参会 Siggraph、Web3D 等世界级图形会议。

3rd SEE Conf

蚂蚁金服体验科技大会，初衷是希望设计与技术能在碰撞中彼此融合，SEE 是 Seeking Experience and Engineering 的缩写，同时 SEE 也代表着“看见”，希望技术能看见设计的价值，也希望设计能看见技术的力量，在彼此看见中互相融合成长，一起让世界更美好。

语雀专栏 | <https://www.yuque.com/seeconf>

SEE Conf 官网 | <https://seeconf.antfin.com/>

参与知乎互动，赢下届门票 | <https://www.zhihu.com/question/363807174>



SEE Conf

体验科技与好的产品

玉伯 (蚂蚁金服 体验技术部负责人)

基于地域文化的设计创新

何人可 (湖南大学设计艺术学院院长)

Ant Design 4.0: 创造快乐工作

林外 (蚂蚁金服 高级体验设计专家)

线丝 (蚂蚁金服 高级创意设计)

决策机构体验科技: 数字驾驶舱

逸达 (蚂蚁金服 前端技术专家)

可言 (蚂蚁金服 高级产品经理)

十喜 (蚂蚁金服 高级体验设计师)

智能可视化体系 AVA

步茗 (蚂蚁金服 数据技术专家、AVA 负责人)

廖鸣 (蚂蚁金服 前端技术专家、DataWizard 负责人)

使用 React 开发小程序 - Remax

边柳 (蚂蚁金服 高级前端工程师、Ant Design 核心贡献者)

Evolution: Serverless For Frontend - 探索下一代 Node 研发模式

天猪 (蚂蚁金服 高级前端专家、Egg.js 核心开发者)

云凤蝶可视化搭建的推导与实现

江木 (蚂蚁金服 高级前端工程师、antd-mobile 核心开发者)

“云”端的语雀 —— 用 JavaScript 全栈打造商业级应用

不四 (蚂蚁金服 高级前端技术专家, 语雀产品技术负责人)

蚂蚁金服 Web 3D 技术探索之路

烧鹅 (蚂蚁金服 前端技术专家、Oasis 3D 引擎负责人)

精雕细琢, 打造极致可视化体验

道为 (蚂蚁金服 高级前端工程师、AntV 核心贡献者)

蚂蚁海外本地化设计

竹摇 (蚂蚁金服 高级体验设计专家)

让价值被发现: 如何在 B 端产品做增长?

覃一 (蚂蚁金服 高级体验设计师)

瀚雅 (蚂蚁金服 高级体验设计师)

围绕应用生命周期的企业级产品设计策略

壹乐 (蚂蚁金服 高级体验设计师)

普惠金融体验设计创新思路: 参与感对话设计

姚维 (蚂蚁金服 体验设计专家)

JCD 思维如何驱动复杂系统设计

今辰 (蚂蚁金服 体验设计专家)

资产的秩序之美: 通过模式化的方法构建设计资产的内在一致性

吾笙 (蚂蚁金服 高级体验设计师)

解放图形化设计生产力 — HiTu

线丝 (蚂蚁金服 高级创意设计)

