
# Advanced Visual Effects in 2D Games

**Viktor Lidholt**
Lead developer – SpriteBuilder

GAME DEVELOPERS CONFERENCE
MOSCONE CENTER · SAN FRANCISCO, CA
MARCH 2-6, 2015 · EXPO: MARCH 4-6, 2015


LEVEL: 2
BOMBS: 0
PAUSE


4666
3

# 2D visual effects

- Sprites
- Distorted polygons
- Particle systems
- Shader effects

# 2D visual effects

- Real-time graphics pipeline
- Effects using normal maps
- Color adjustments
- Blur & bloom
- Putting it all together

# Tools for building effects

- Creating assets
- Previewing effects
- Using good tools aids experimentation

# 2D lighting


Refraction

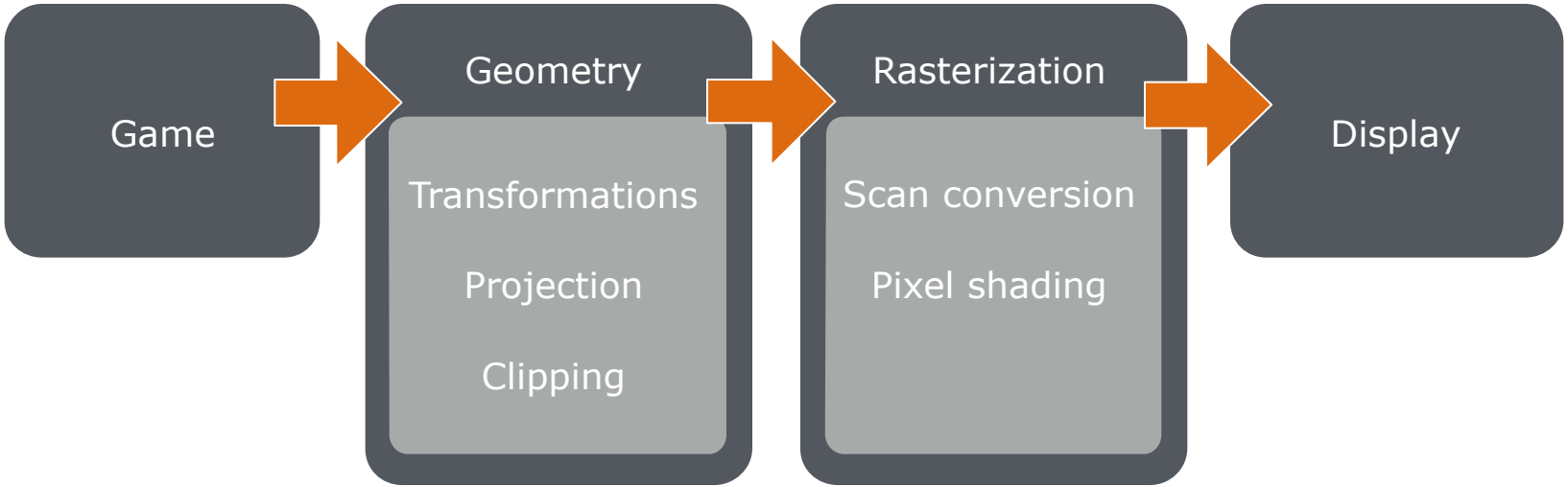

Reflection


Blur


Bloom


Color adjustments

# Graphics pipeline



# How effects are rendered

- Using sprites or polygons
  - Particle systems
  - Motion streak effects
- Using shaders on the graphics card
  - Pretty much any effect

# What is a shader?

- Programs compiled on the graphics card
- Platform dependent
  - GLSL / OpenGL
  - Metal shading language
  - HLSL / DirectX
- C or C++ like language

# What is a shader?

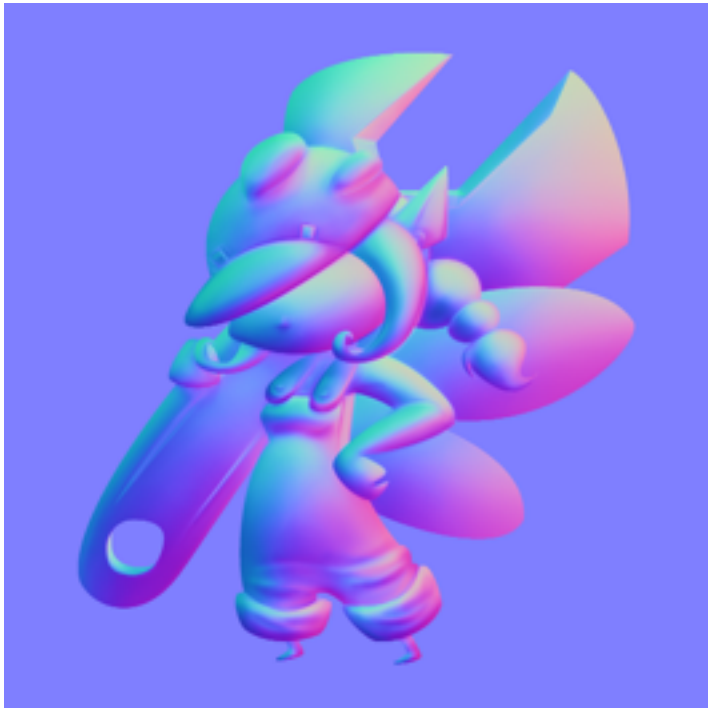
- Introduced in OpenGL ES 2
- Not supported on older hardware
- Performance varies with graphics card
- Requires testing on a multitude of devices

# What is a shader?

- Vertex shaders
  - Modifies the location of vertices
- Pixel (fragment) shaders
  - Computes the color of each rendered pixel
  - Can sample the environment
  - Can use multiple textures

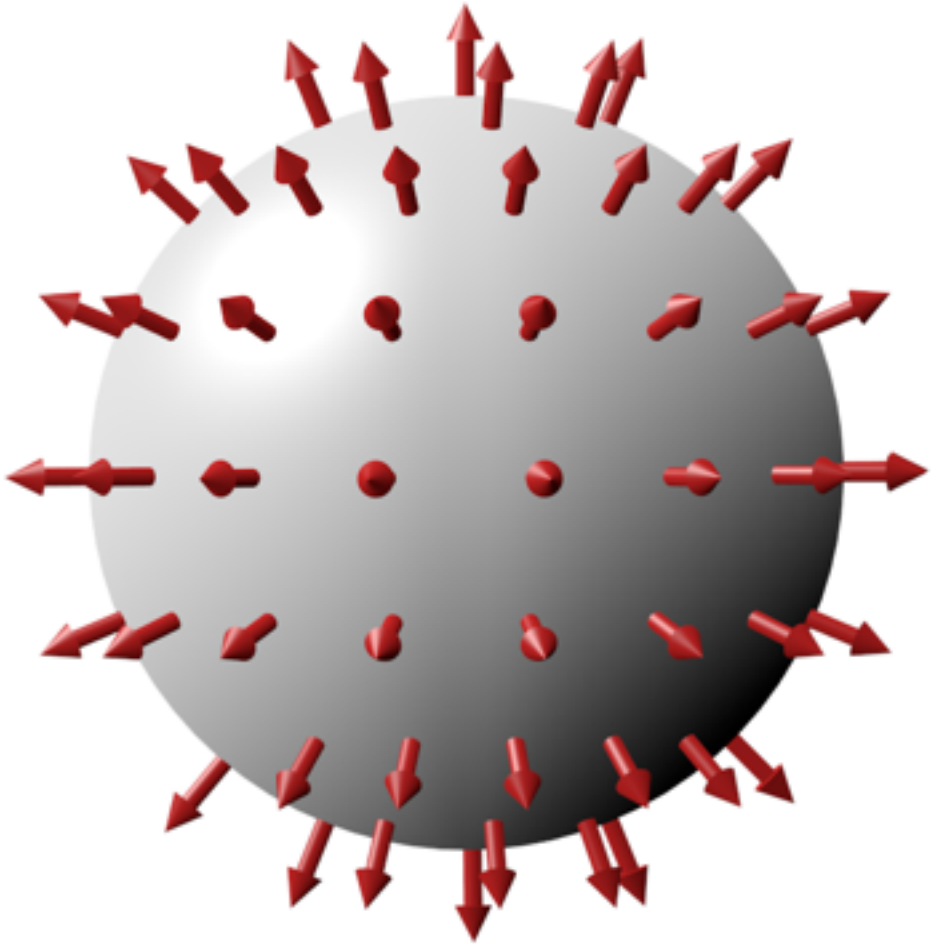
# Normal maps

"A normal map is an image where the RGB components correspond to the X, Y, and Z coordinates, respectively, of the surface normal."



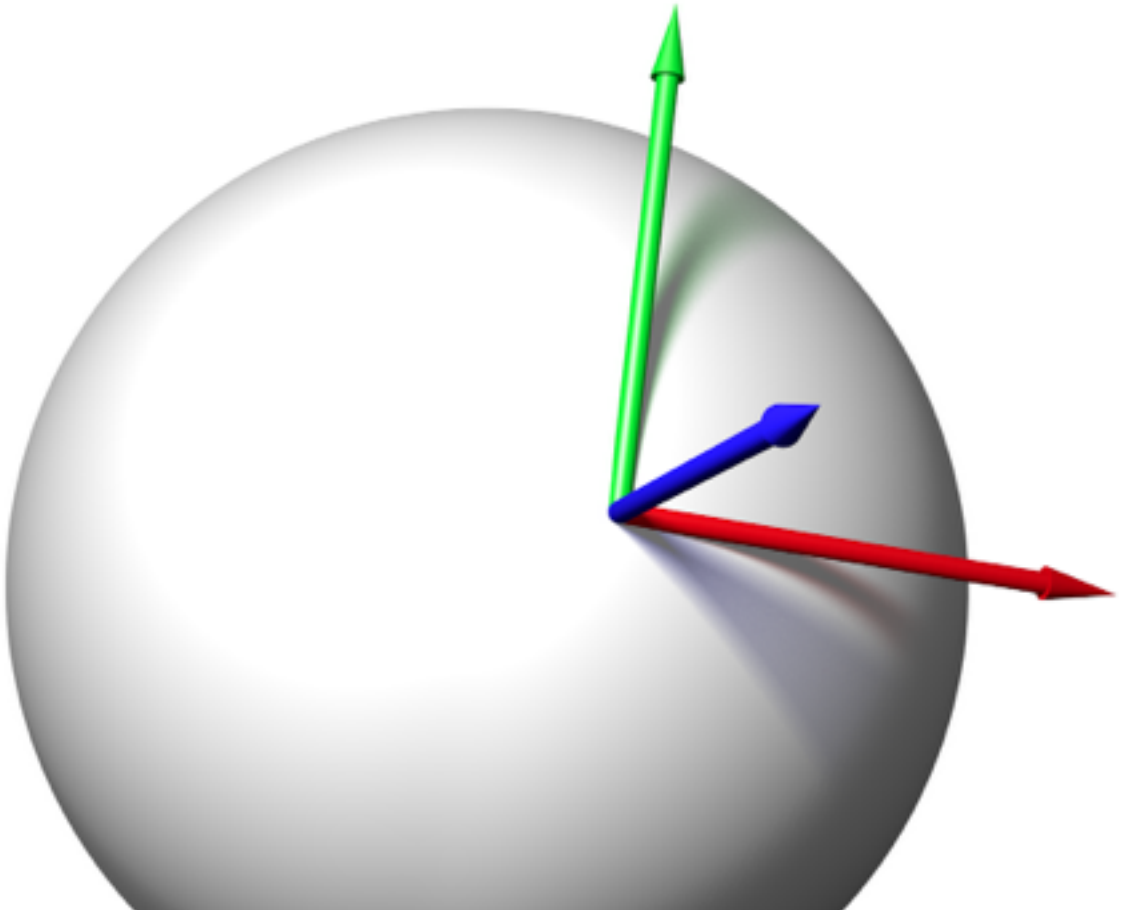
# What is a normal?

A normal is a vector that is perpendicular to a given object’s surface



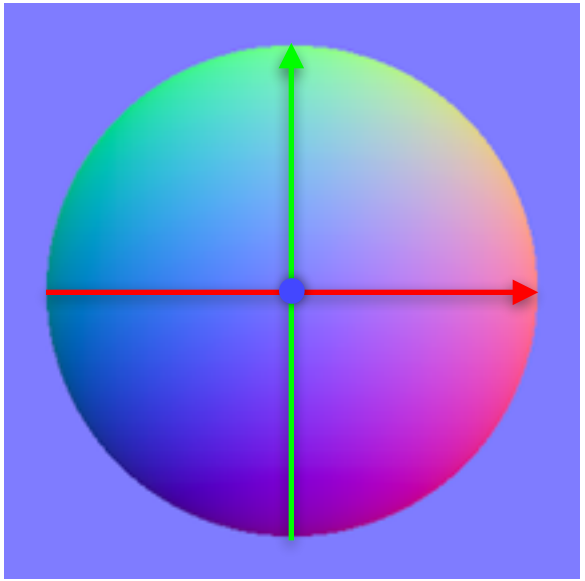
# What is a normal?

- Any vector in 3D space can be decomposed into three components (x,y,z)
- These components can be mapped to the RGB values in an image



# Normal maps

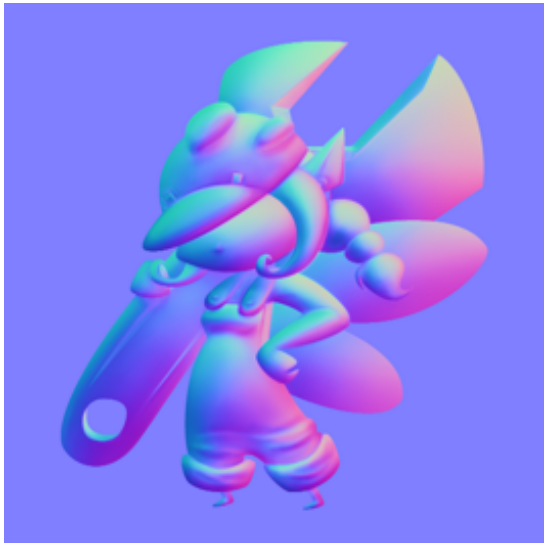
- A normal map depicts the normals of a 3D object as viewed from a specific direction
- Each pixel’s color corresponds to the surface normal at that position



# 2D lighting

- Fast to render on any modern hardware
- Relatively easy to implement
- The real problem is the complexity of building the normal maps

Diffuse map



Normal map

Lighted sprite


4666
3

# Tweaking lighting

- Ambient light
  - Base lighting
- Specular light
  - Highlights or gloss
  - Color & intensity can be modified
  - Custom specular texture maps

Diffuse Ambient Specular

# Making it look great

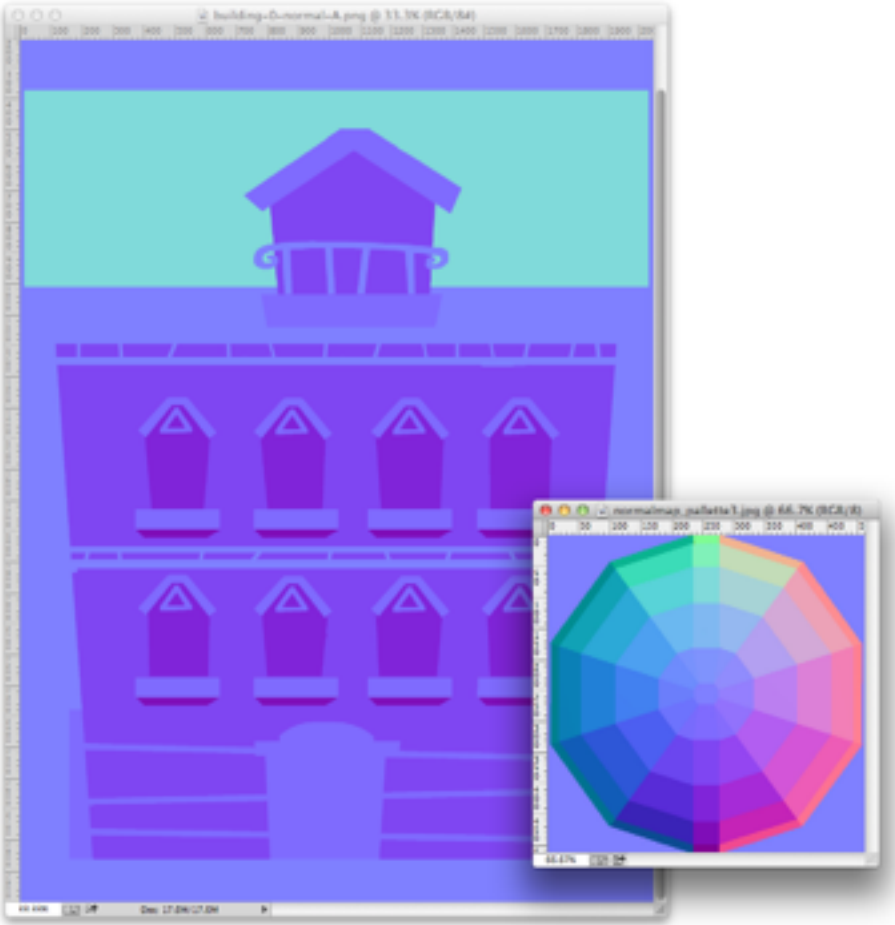
- Requires experimentation
- Using tools make the process faster
- Find or make the right tools for you

# Creating normal maps

- Manually drawn
- Generate in 3D application
- Build from height map
- Specific tools
- Blending normal maps

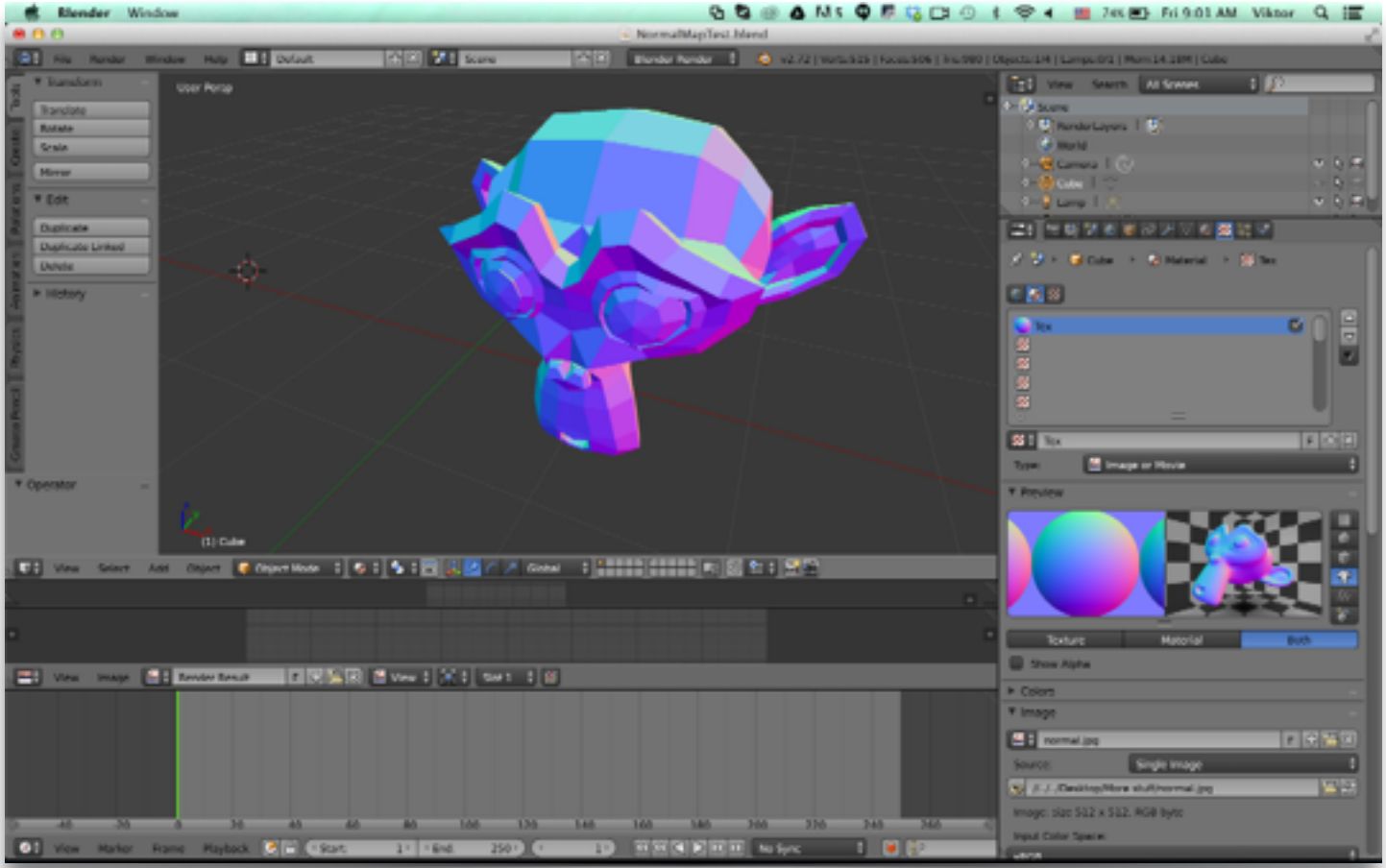
# Manually drawn

- Use the normal sphere to pick colors
- Good for sharp edges
- Anti-alias can have unexpected effects
  - Be careful while drawing
  - Be careful while scaling down



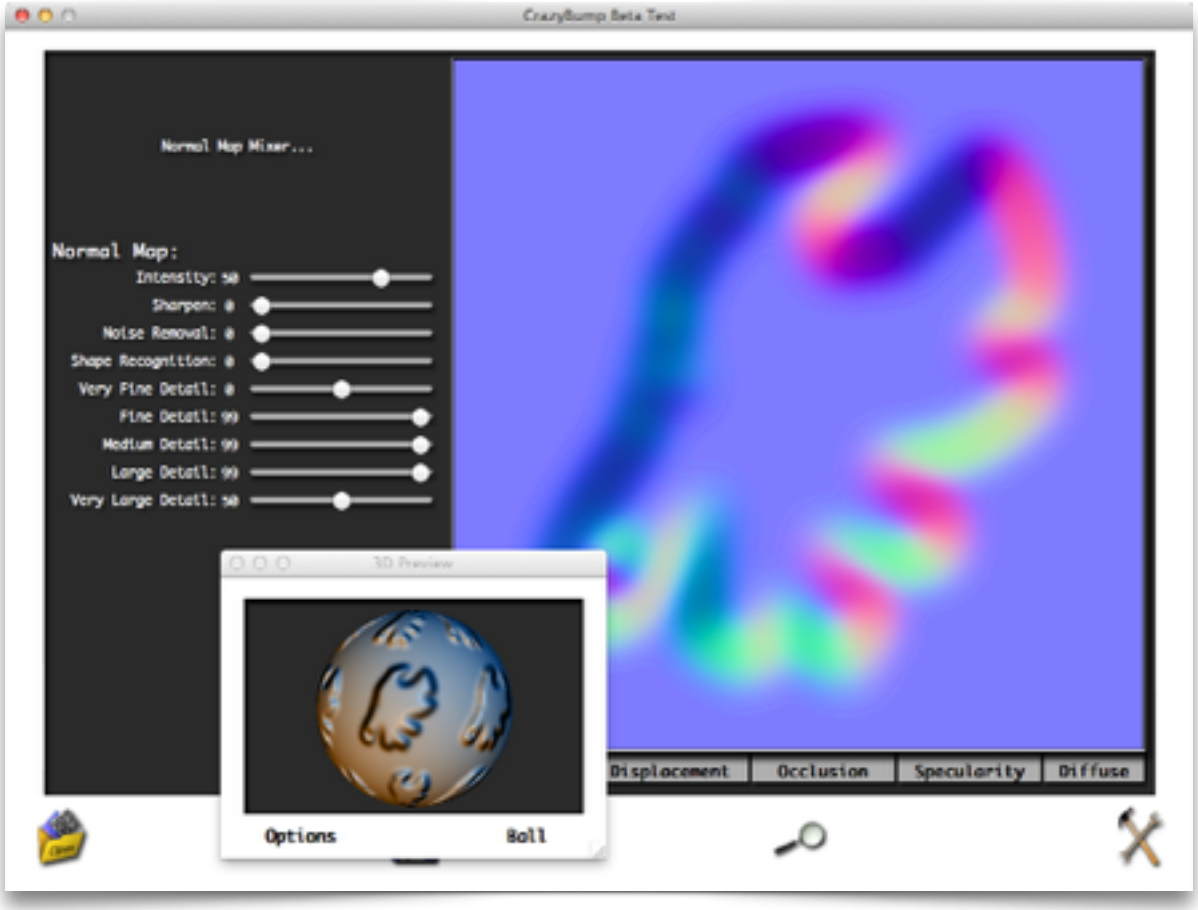
# Generate in 3D application

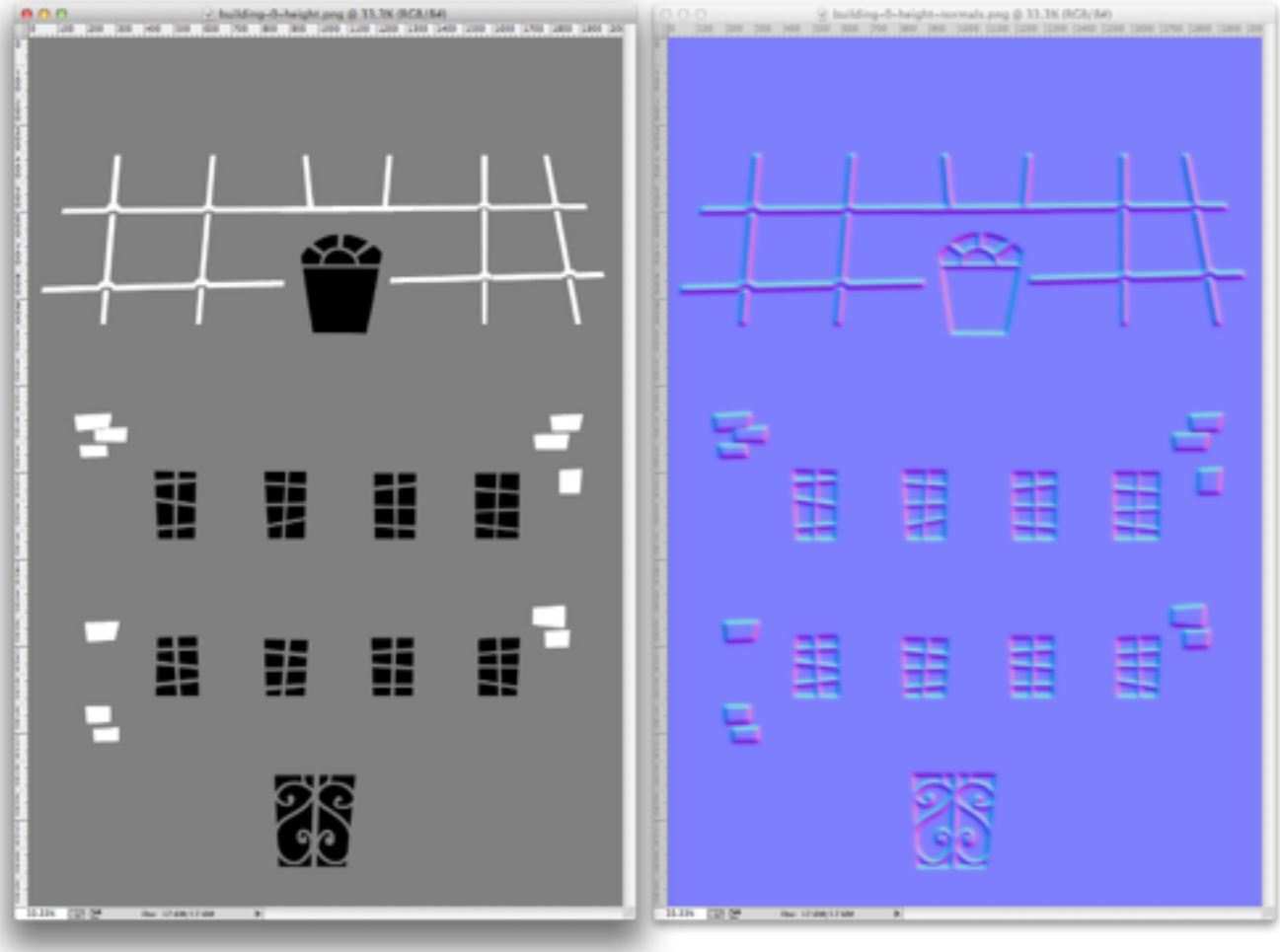
- Good, predictable results
- Easy to setup in 3D program
- Requires 3D models of your art


Blender Window
User Persp
Operator
Transform
Translate
Rotate
Scale
Mirror
Edit
Duplicate
Duplicate Linked
Delete
History
Tex
Type:
Image or Movie
Preview
Texture
Material
Both
Show Alpha
Colors
Image
normal.jpg
Source:
Single Image

# Build from height map

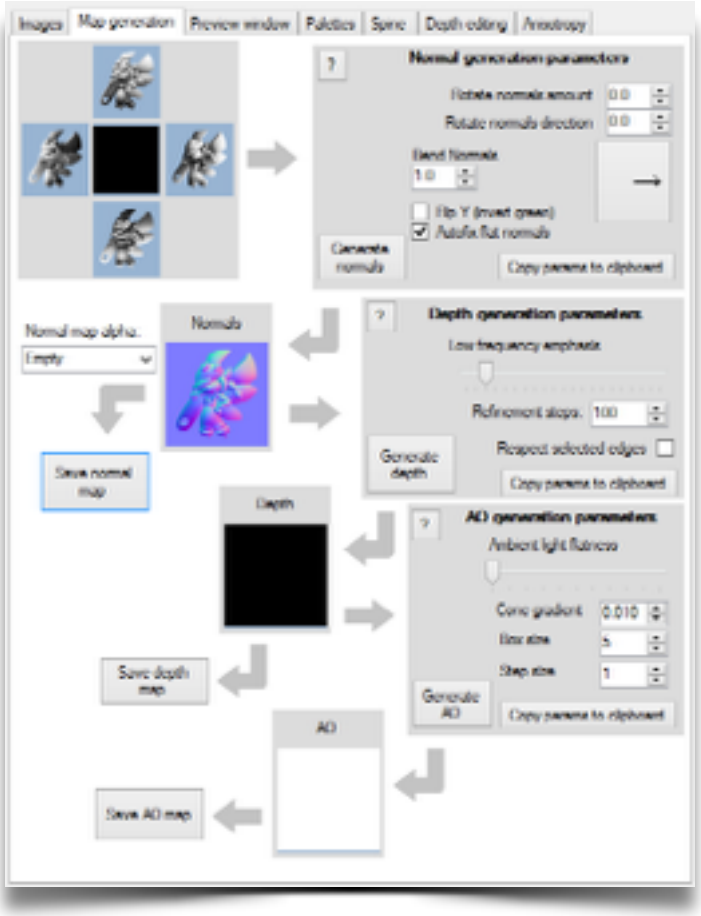
- Height maps can be manually drawn
- Good for details
- Results may need tweaking

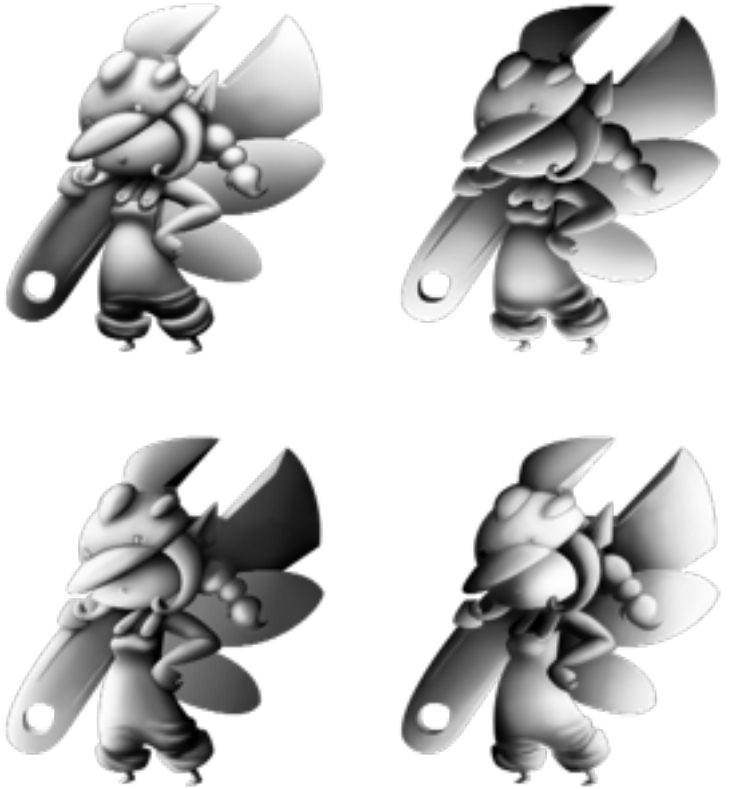

CrazyBump Beta Test
Normal Map Mixer...
Normal Map:
Intensity: 50
Sharpen: 0
Noise Removal: 0
Shape Recognition: 0
Very Fine Detail: 0
Fine Detail: 99
Medium Detail: 99
Large Detail: 99
Very Large Detail: 50
3D Preview
Displacement
Occlusion
Specularity
Diffuse
Options
Ball

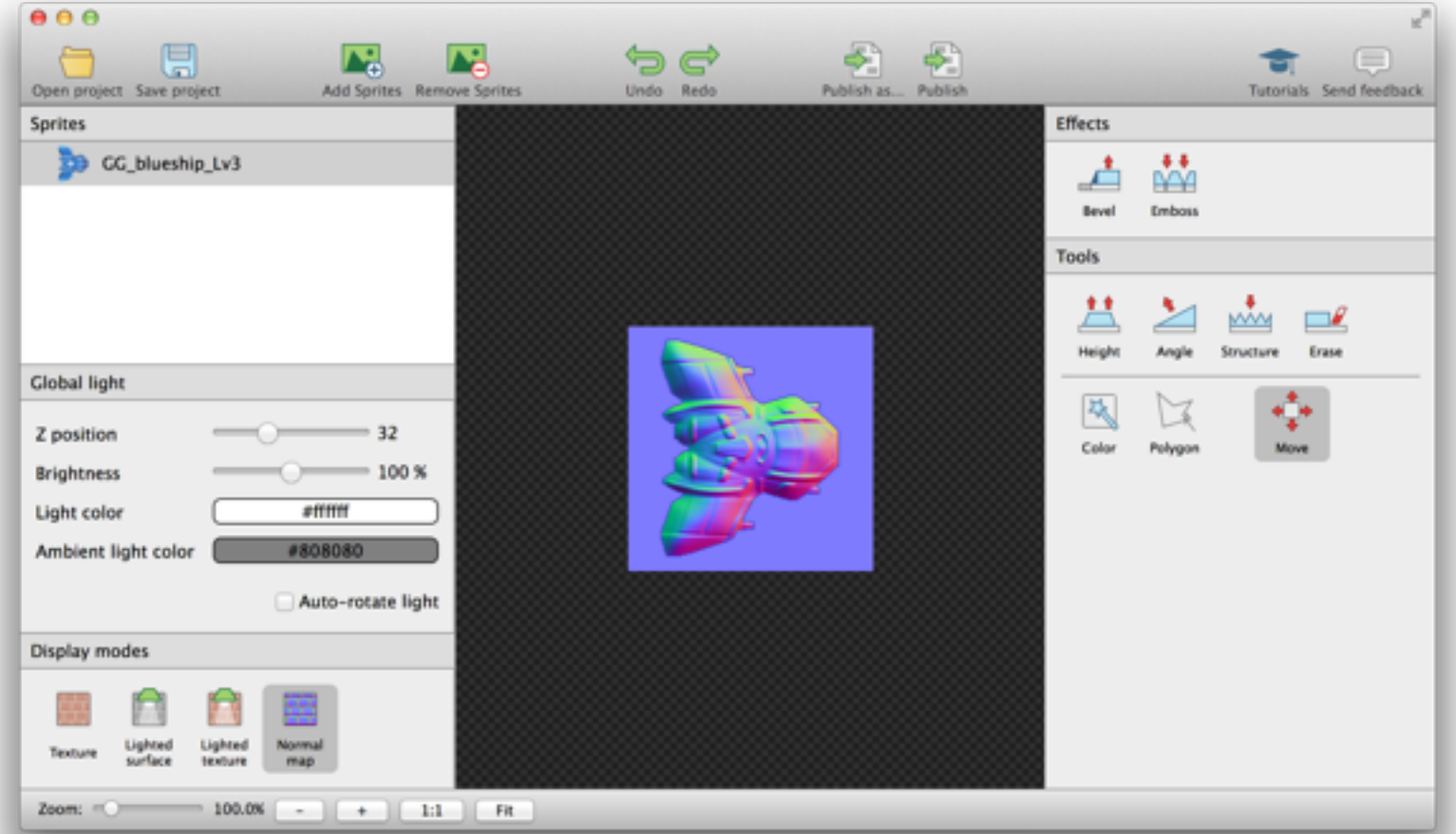


# Specific tools

- SpriteLamp
  - Great results for organic shapes
  - Requires a lot of extra art
- SpriteIlluminator
  - Drawing program for normal maps
  - Some sprites can be tricky

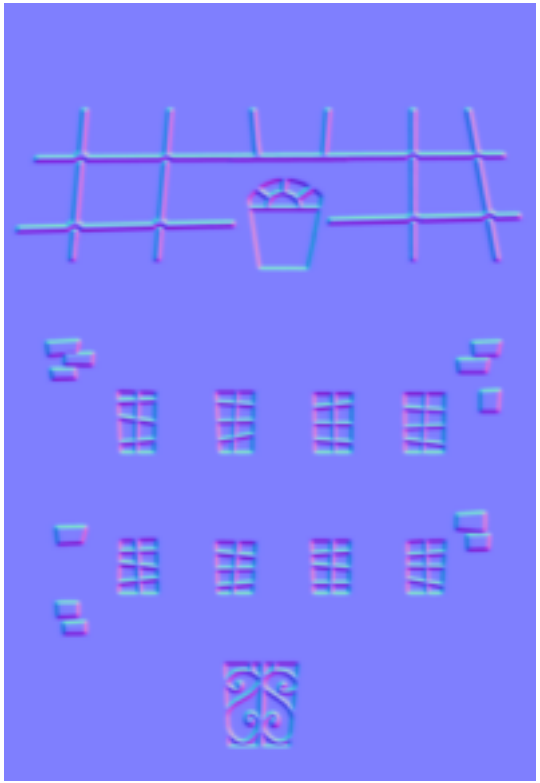

Images
Map generation
Preview window
Palettes
Sprite
Depth editing
Anisotropy
Normal generation parameters
Rotate normals amount
Rotate normals direction
Bend Normals
Flip Y (invert green)
Autofix flat normals
Generate normals
Copy params to clipboard
Normal map alpha:
Empty
Normals
Save normal map
Depth generation parameters
Low frequency emphasis
Refinement steps:
Respect selected edges
Generate depth
Copy params to clipboard
Depth
AO generation parameters
Ambient light fluttress
Cone gradient
Step size
Generate AO
Copy params to clipboard
Save depth map
AO
Save AO map




Open project
Save project
Add Sprites
Remove Sprites
Undo
Redo
Publish as...
Publish
Tutorials
Send feedback
Sprites
GG_blueship_Lv3
Global light
Z position
32
Brightness
100 %
Light color
#ffffff
Ambient light color
#808080
Auto-rotate light
Display modes
Texture
Lighted surface
Lighted texture
Normal map
Effects
Bevel
Emboss
Tools
Height
Angle
Structure
Erase
Color
Polygon
Move
Zoom:
100.0%
-
+
1:1
Fit

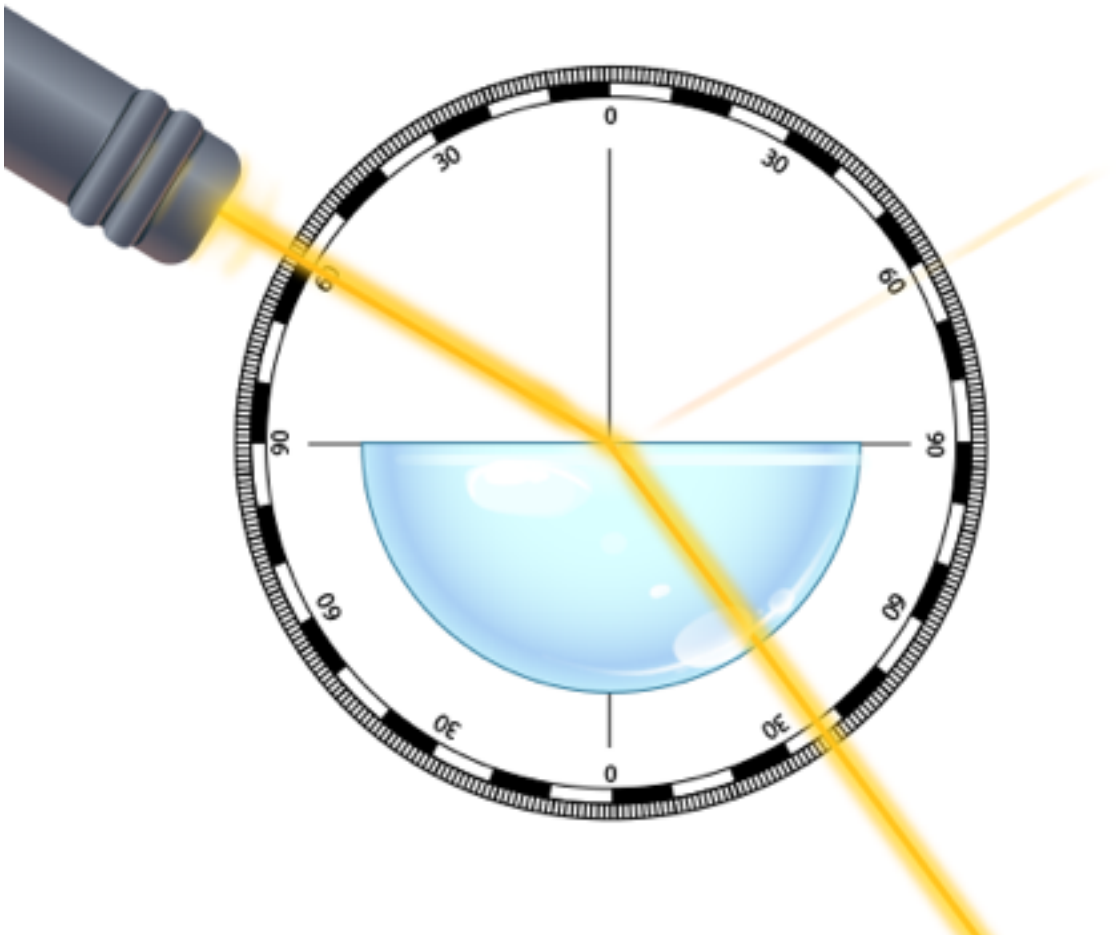
# Blending normal maps

- Simply blending / adding colors won’t give expected results
- Special purpose tools available

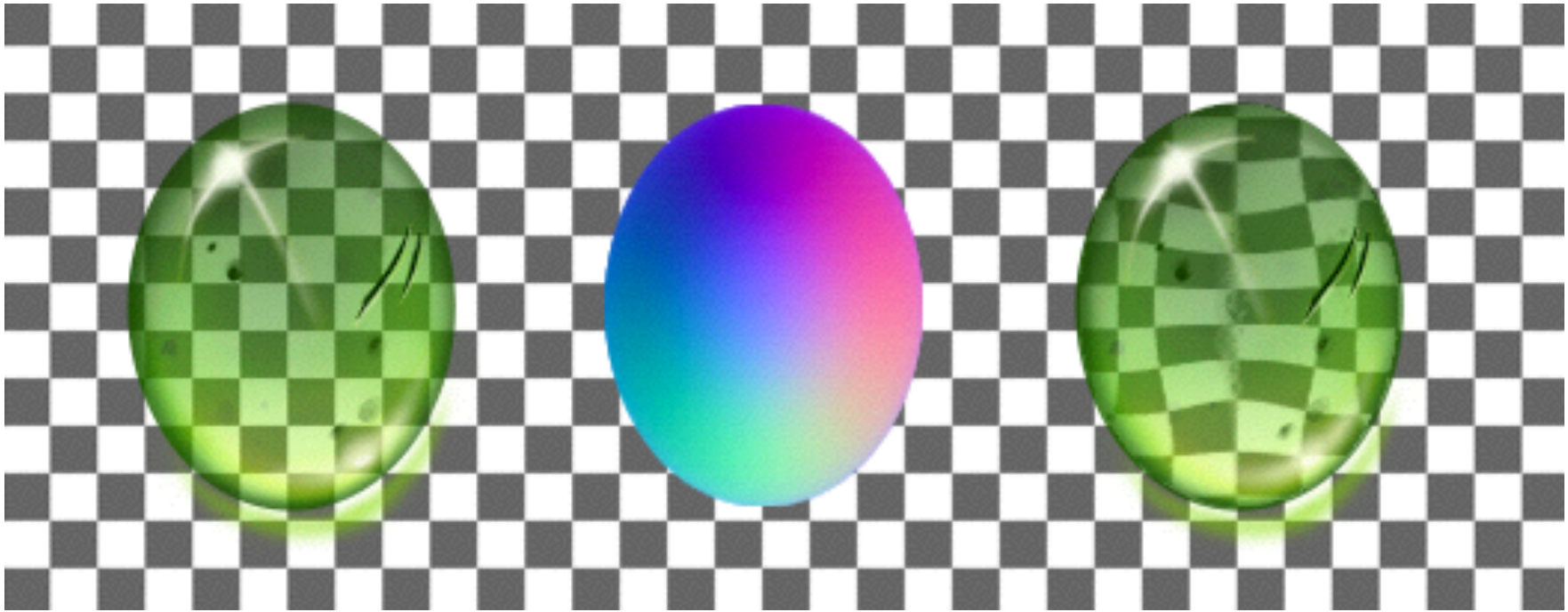


# Refraction effects

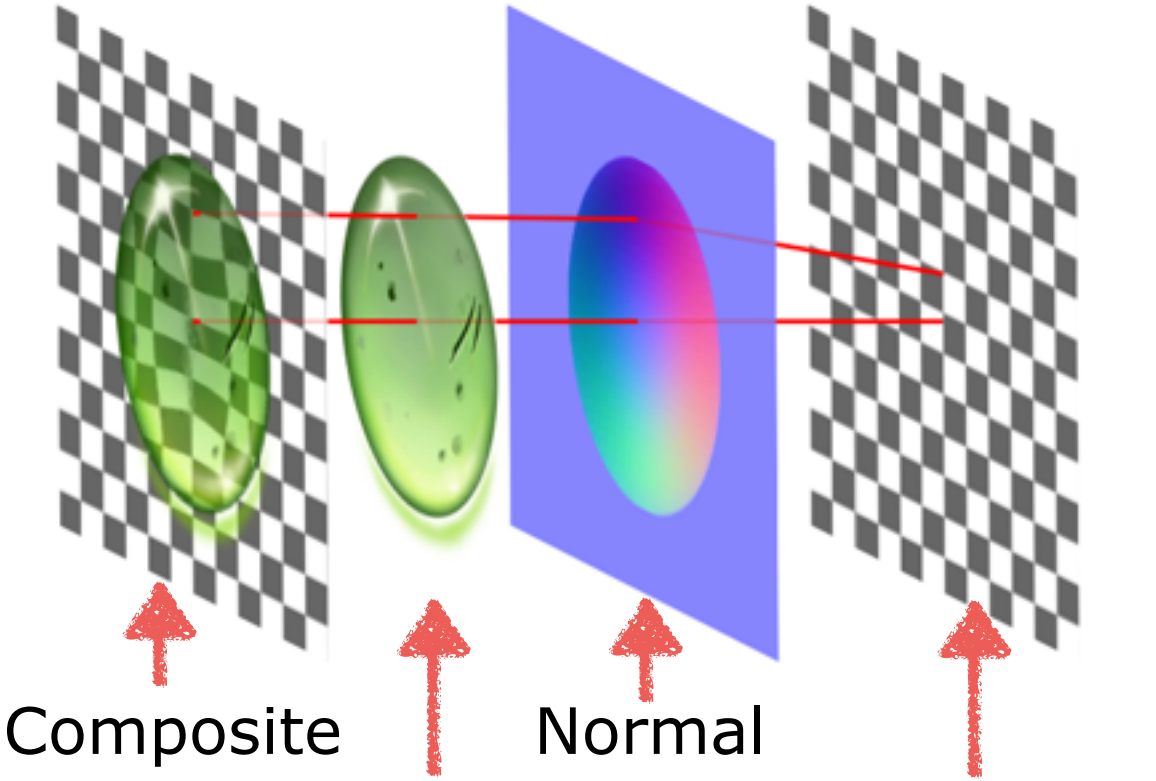
- Simulates light breaking through transparent materials
- Uses normal maps in combination with environment (refraction) maps


0
30
60
90
60
30
0
30
60
90
60
30


Refraction



Diffuse map Normal map Refraction

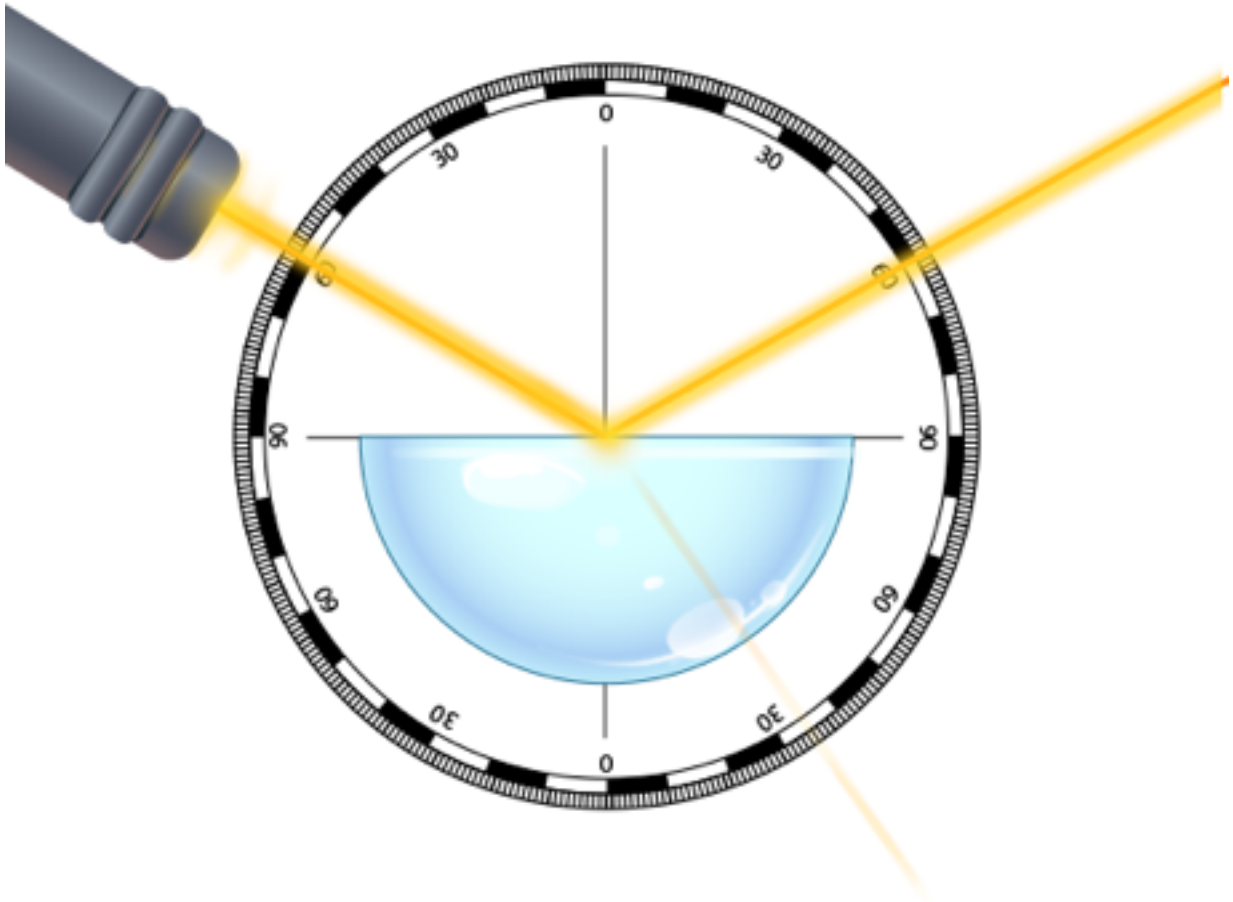

Composite
Diffuse
Normal
Environment

# Environment map

- Single sprite
  - Fast
  - Cannot render intermediate objects
- Render texture
  - Slower
  - Can have animations in the environment

# Reflection effect

- Much like refraction, but simulates reflection
- Uses normal maps in combination with environment (reflection) maps


0
30
60
90
30
60
0
30
60
90
30
60
0


TAP TO PLAY
JACKPOT
1000
Reflection


Reflection

# Tips on normal map effects

- Combine refraction and reflection to create glass-like effects
- Environment maps can be moved or rotated to appear more dynamic
- Effects doesn’t need to be perfect to look good

# Color adjustments

- Reduce number of assets
- Highlight game elements
- Improve animations
- Transitions

# Color adjustments

- Saturation
- Contrast
- Brightness
- Hue

Saturation

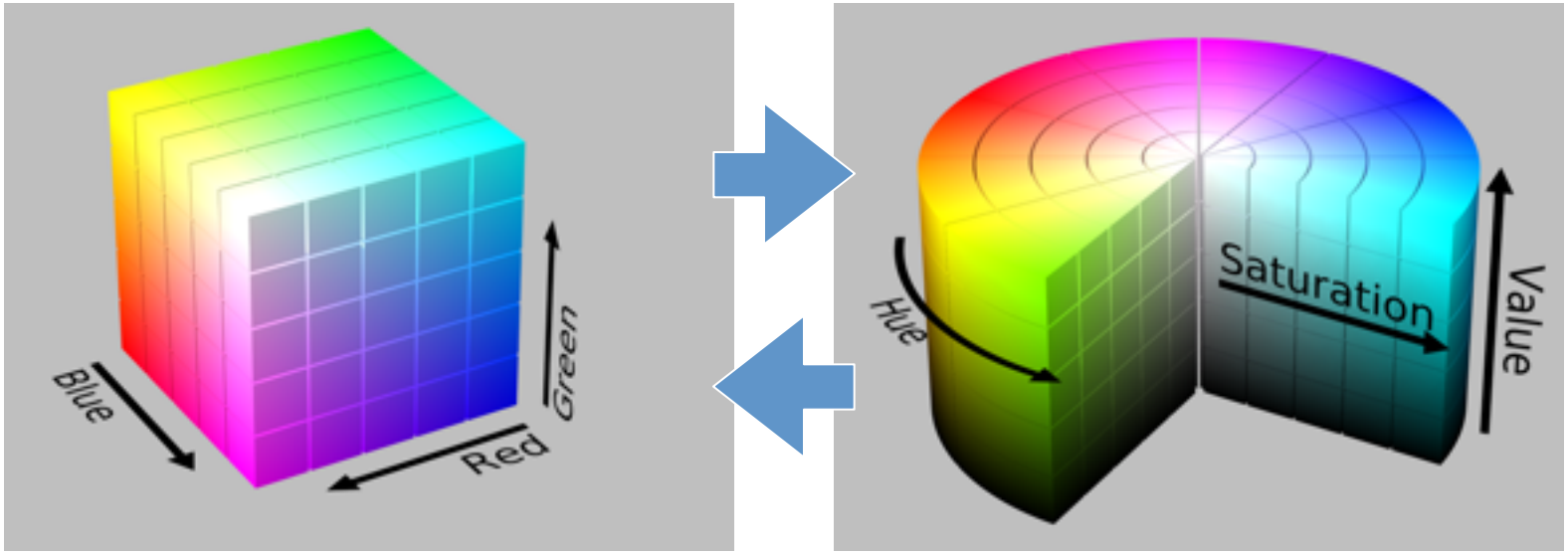
Contrast

Brightness

Hue

# Color adjustments

- Saturation, Contrast & Brightness can all be done in the RGB color space
- Shifting the hue requires conversions between color spaces



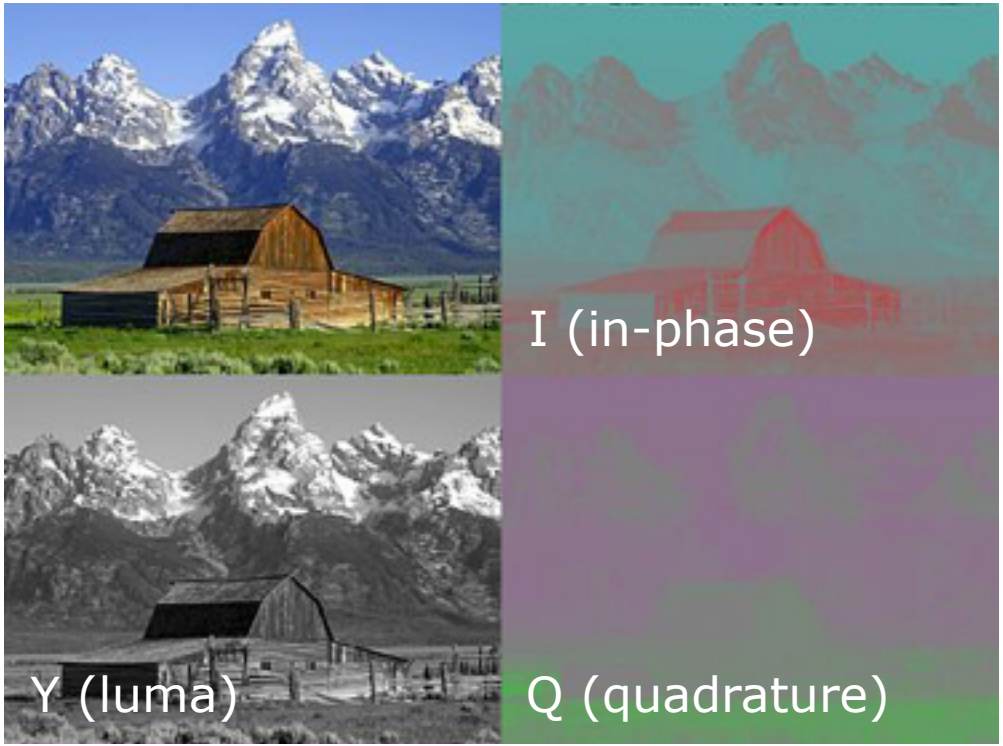

RGB HSV

# Shifting hue

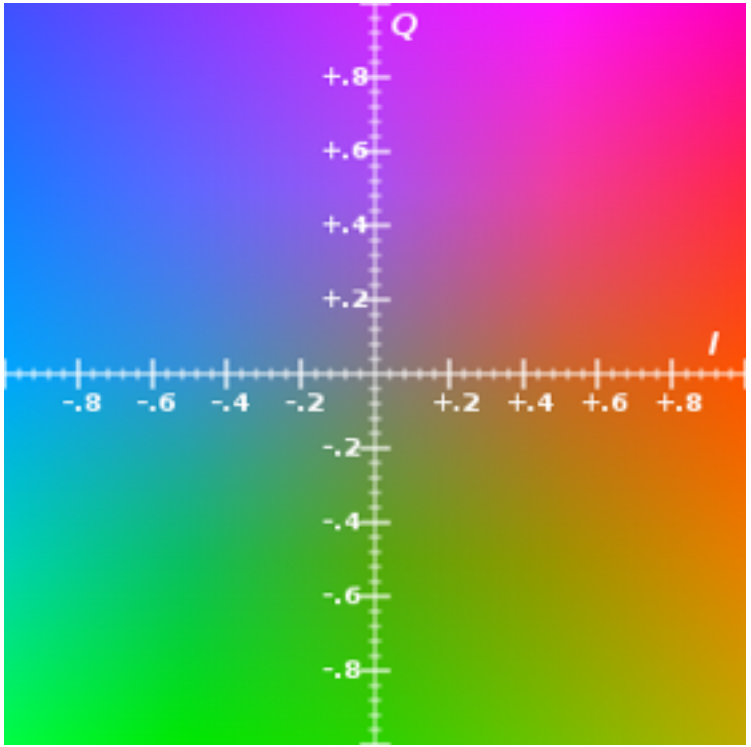
- Converting to HSV is slow
- Hue shift can be approximated in YIQ color space
- Shifting hue using YIQ is twice as fast, but not perfect

# YIQ colorspace

- Used by NTSC color TV system
- Luma, in-phase & quadrature
- Conversion to/from RGB is done by a simple matrix multiplication
- Can shift the hue by rotating around the luma axis




YIQ channels




Color cube at luma = 0.5

## HSV

Preserves saturation

## YIQ

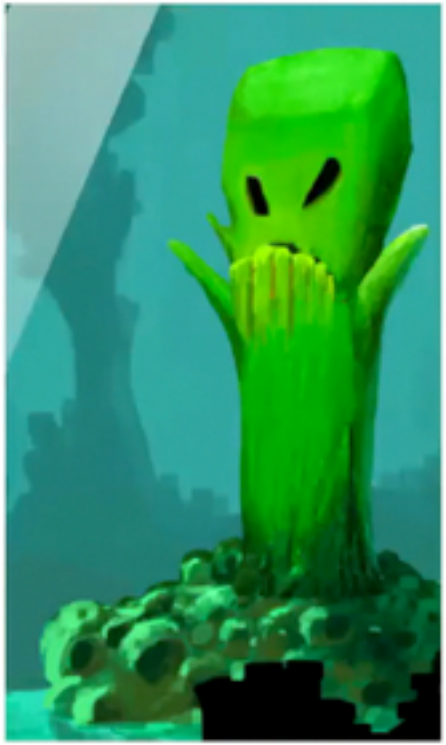
Preserves luminance

# Reducing number of assets

- Color sprites or part of sprites, instead of adding more textures saves memory
- Can impact performance
  - Color adjustments are slower to render than plain sprites
  - Adding different shaders break sprite batching

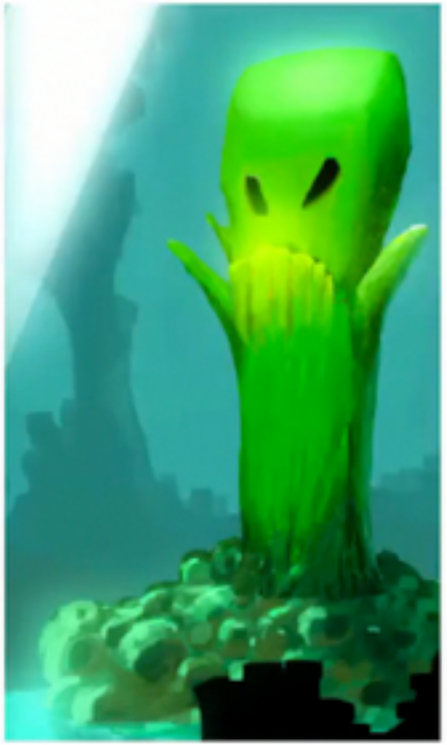
# Blur & bloom

- Blur
- Bloom
- Drop shadow
- Glow



Original

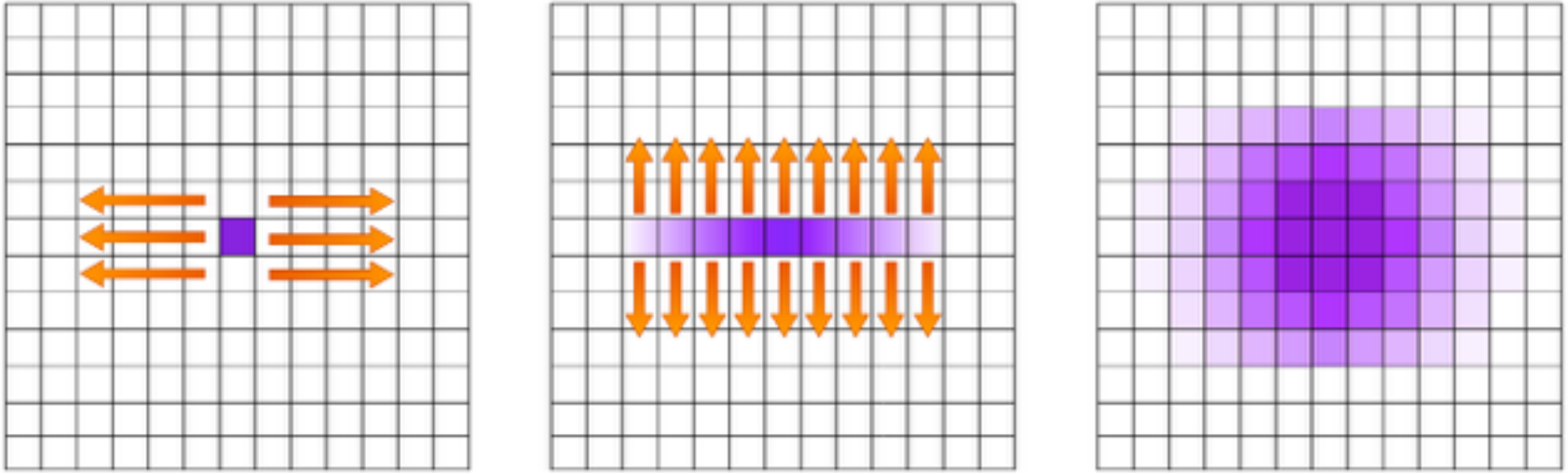
Blur



Bloom

# Blur

- Doing correct blurs are very expensive
- Can be simplified with multiple render passes
- Different implementations are faster on different devices



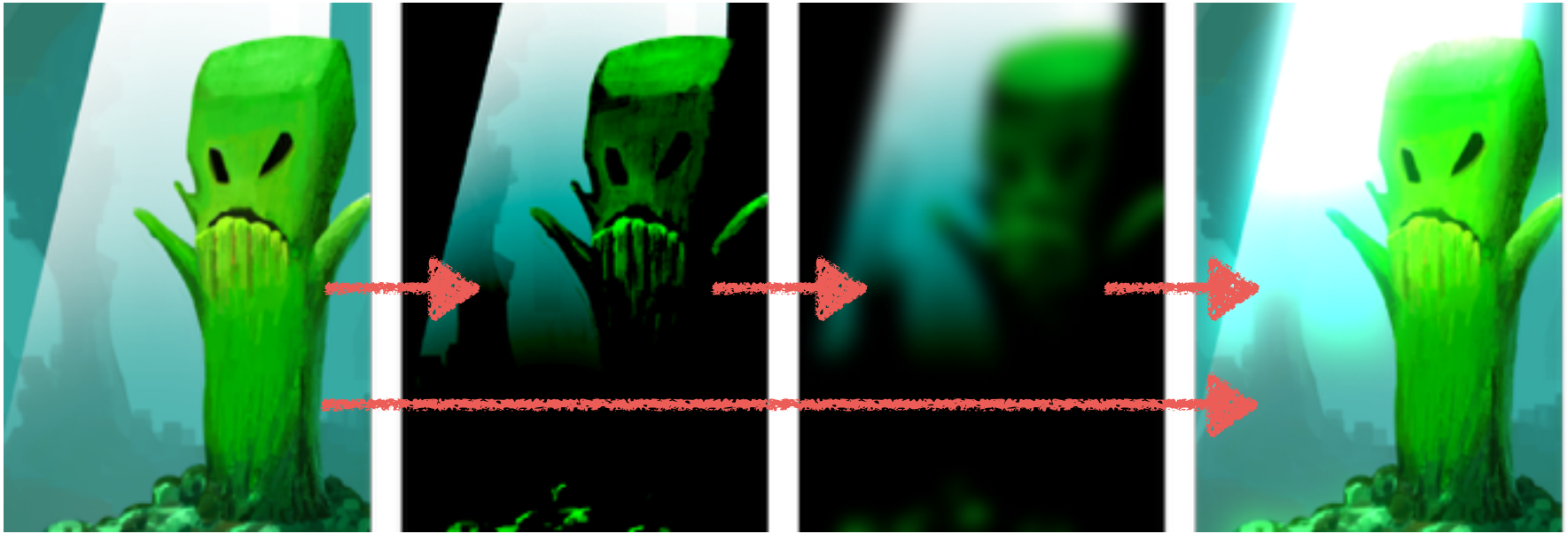
Blur using multiple render passes

# Blur optimizations

- Use pre-rendered images
  - Dissolve/blend between can simulate dynamic results
- Buffer the results in a render texture
  - Useful when the blurred areas are not dynamic

# Bloom

- Feathers of light extending from brightly lit areas
- Created by combining blur with filtering
- Expensive to render



Original Filtered Blurred Added

# Putting it all together

- Tweaking performance
- Combining effects

# Good toolsets for effects

- Allows visual exploration
- Makes it easier to tweak performance
- Minimizes number of iterations
- Can be used by designers

# Tweaking performance

- Processing power varies greatly among mobile devices
- Not all effects are essential for game-play
- Blur, bloom, refraction & reflection are particularly expensive

# Test performance

- Determine device type
  - Thousands of devices
- Render graphics offscreen
  - May not reflect speed perfectly
  - Takes time

# Test performance

- Load assets before test
- Start with basics, add in more effects
- Measure frame rate
- Can be done in about one second

# Demo

# Mentioned tools

- SpriteBuilder
- CrazyBump
- Normal Mixer
- Sprite Lamp
- SpriteIlluminator

# More info and resources

viktorious.com/gdc

- Slides from presentation
- Links to all tools
- Links to more tutorials

@viktorlidholt