Game Technology

Optional Lecture 15 – 13.2.2015
Scripting

TECHNISCHE UNIVERSITÄT DARMSTADT

Dr.-Ing. Florian Mehm
Dipl-Inf. Robert Konrad

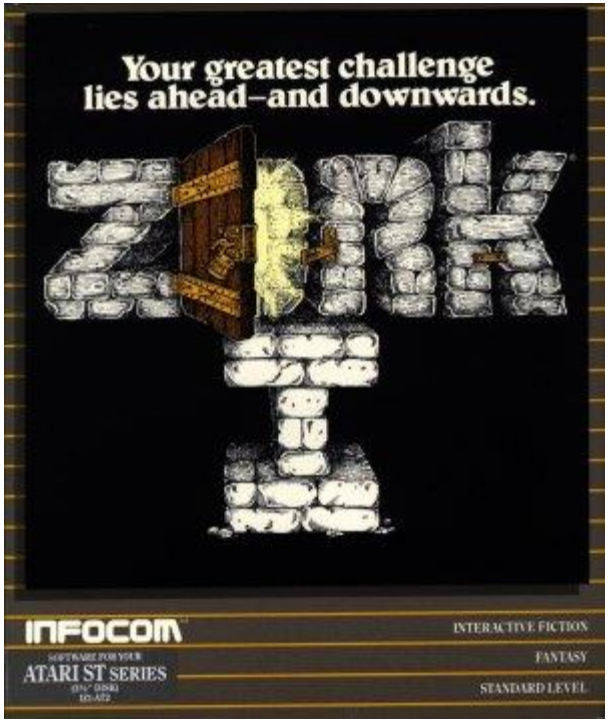
Prof. Dr.-Ing. Ralf Steinmetz
KOM - Multimedia Communications Lab

History

Zork Implementation Language (ZIL)

- 1979
- Created by Infocom to facilitate the creation of interactive fiction titles
- Compiles to code for a virtual machine → Z-Machine

```
<ROOM LIVING-ROOM
      (LOC ROOMS)
      (DESC "Living Room")
      (EAST TO KITCHEN)
      (WEST TO STRANGE-PASSAGE IF CYCLOPS-FLED ELSE
           "The wooden door is nailed shut.")
      (DOWN PER TRAP-DOOR-EXIT)
      (ACTION LIVING ROOM-F)
      (FLAGS RLANDBIT ONBIT SACREDBIT)
      (GLOBAL STAIRS)
      (THINGS <> NAILS NAILS-PSEUDO)>
```



History

AGI – Adventure Game Interpreter

- 1984
- Created by Sierra On-Line for graphical adventure games
- First used fully in King‘s Quest
- Superseded by SCI – Sierra Creative Interpreter

```
if (said("look","door")) {
  if (posn(ego,0,120,159,167)) {
    print("These doors are strongly
built
      to keep out unwanted visitors.");
  }
  else {
    print("You can't see them from
      here.");
  }
}
```

History

TECHNISCHE UNIVERSITÄT DARMSTADT

SCUMM – Script Creation Utility for Maniac Mansion

- 1987
- Created by Lucasfilm Games for… Maniac Mansion

```
cut-scene {
    ...
    actor nurse-edna in-room edna-bedroom
at 60,20
    camera-follow nurse-edna
    actor nurse-edna walk-to 30,20
    wait-for-actor nurse-edna
    say-line nurse-edna "WHATS'S YOUR
POINT ED!!!"
    wait-for-talking nurse-edna
    ...
}
```

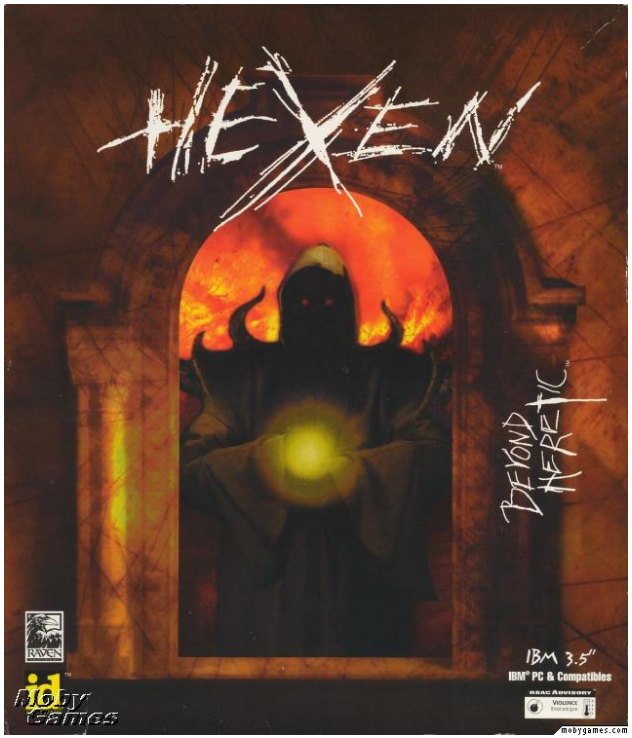
History

TECHNISCHE UNIVERSITÄT DARMSTADT

Action Code Script

- 1995
- Created by Raven Software for Hexen, extending the original Doom engine
- Allowed scripting events during a level

```
SCRIPT 4 (void)
{
  suspend;
  suspend; // The statements "absorb" the
effect of the two first toggles, whichever
they are
  ambientsound("Chat",127);
  printbold(s:"SEQUENCE COMPLETED!");
}
```



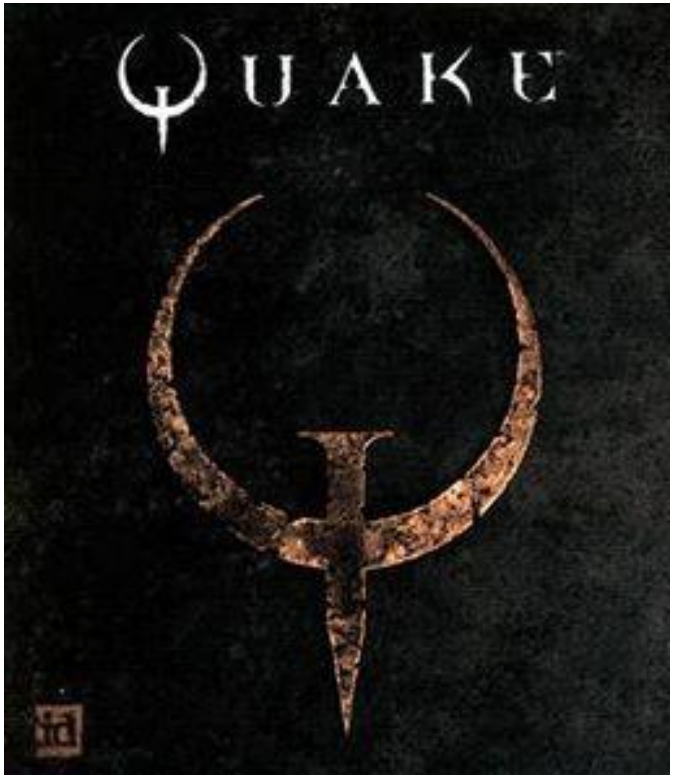
History

TECHNISCHE UNIVERSITÄT DARMSTADT

QuakeC

- 1996
- Created by id Software to control Quake

```
void (float v) ai_berserk =
{
  if (self.health > 25)
  {
    ai_run (v);
  }
  else
  {
    ai_run (v * 1.5); // adjust to your
taste
    self.nextthink = time + 0.075; //
adjust to your taste
  }
};
```



History

TECHNISCHE UNIVERSITÄT DARMSTADT

UnrealScript

- Developed in 1998 for the first Unreal

Can extend the class hierarchy of Unreal

```
class HappyHUD extends HUD
        config(Game);

//VARS
var String joyMessage;
var FONT joyFont;

…
function DrawHUD()
{

     local string StringMessage;

     //projection of ray hit location
  local vector HitLocation, HitNormal;

  //get origin and dir
  Canvas.DeProject(MousePosition, WorldOrigin, WorldDirection);

  StringMessage = "MouseX" @ MousePosition.X @ "MouseY" @ MousePosition.Y;
```

Advantages of Scripting Languages

TECHNISCHE UNIVERSITÄT DARMSTADT

Designer-Friendly

- Designers, non-programmers are enabled to work on the game directly, without needing programmer resources (ideally…)
- Quickly change values, …

Easy to learn

- Often reduced complexity compared to C++ or similar languages
- Often no memory management, pointers, …

Adaptable

- Many scripting languages are flexible
- E.g. Ruby or Lua allow adapting the language itself, e.g. to create a domain-specific language

Advantages of Scripting Languages

TECHNISCHE UNIVERSITÄT DARMSTADT

Concurrency

- Coroutines
- Functions that can be interrupted and continued

No compilation

- No additional time during compiling the game (engine)
- Can be switched during runtime
- Downside: Often slower than compiled code

Mod-support

- Allows players to change the game using the scripting language
- Increases shelf-life

Runtime vs. Data Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

Data-definition languages

- Create data structures that control the game engine
- E.g. LISP-dialect used by Naughty Dog

```
(define-export *player-start*
(new locator
:trans *origin*
:rot (axis-angle->quaternion *y-axis* 45)
))
```

Runtime scripting languages

- Control the game during runtime
- All examples in the history slides are of this kind

Common language properties

TECHNISCHE UNIVERSITÄT DARMSTADT

Interpreted

- Flexibility, portability and rapid iteration
- Virtual machine → port the VM to port the scripts

Lightweight

- Simple, low memory footprints

Support for rapid iteration

- Quicker turnaround time
- See changes immediately/after a restart

Convenience

- Tuned for the purpose in the game

Textual Languages

TECHNISCHE UNIVERSITÄT DARMSTADT

All languages we have seen so far

Special case: Natural-language Programming

Can be found in Inform 7 (interactive fiction tool)


```
The shower is here. It is fixed in place. "Opposite the
  mirror is the shower, which is closed." The description of
  the shower is "When it's open, you get in it to take a
  shower. Right now it's closed, keeping you from using it."

Instead of opening or entering the shower, say "It is locked
  down until after the ship makes its jump to hyperspace."
```

Textual Languages

TECHNISCHE UNIVERSITÄT DARMSTADT

All languages we have seen so far

Special case: Natural-language Programming

Can be found in Inform 7 (interactive fiction tool)


```
The shower is here. It is fixed in place. "Opposite the
 mirror is the shower, which is closed." The description of
 the shower is "When it's open, you get in it to take a
 shower. Right now it's closed, keeping you from using it."

Instead of opening or entering the shower, say "It is locked
 down until after the ship makes its jump to hyperspace."
```

http://inform7.com/

Lua

TECHNISCHE UNIVERSITÄT DARMSTADT

http://www.lua.org/

Development started in 1993 at Pontifical Catholic University of Rio de Janeiro

Small language core

„Events“

- Fired when operators/functions are called, …
- Native code can register to handle them

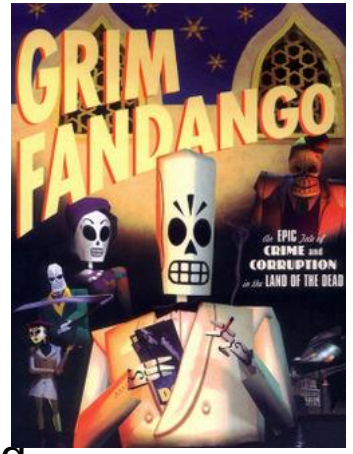
Tags

- Code called when events are fired
- Allow Lua behaviour itself to be changed

Lua Example

TECHNISCHE UNIVERSITÄT DARMSTADT

Used in Grim Fandango



http://www.lua.org/wshop05/Mogul.pdf

- – Dialogue
- – Puzzle logic
- – UI/controls
- – Menus
  - Engine handles only animations, backgrounds, sound, rendering, choreography, etc etc etc... But those aren't Grim Fandango

Python

TECHNISCHE UNIVERSITÄT DARMSTADT

https://www.python.org/

Development started in 1989 by Guido van Rossum as a hobby project

Easier to learn for non-programmers than other languages

Disadvantages: Large size and speed

- Relies on hash table lookups

Eve Online server almost completely written in Stackless Python

Visual Languages

TECHNISCHE UNIVERSITÄT DARMSTADT

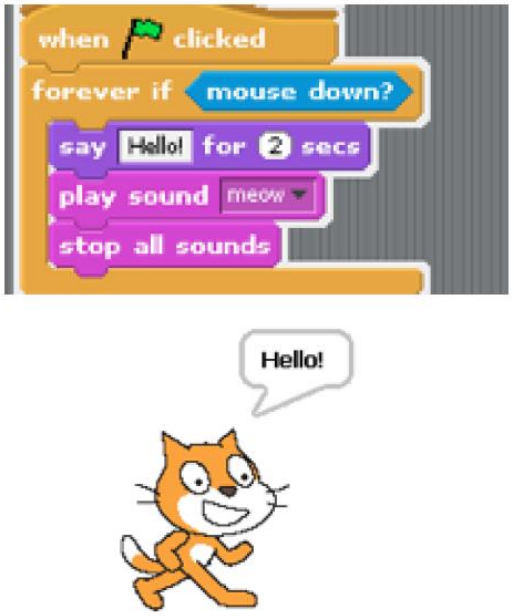
Emerging trend in game tools

Designer-friendly, easy to debug/visualize scripts

Can become complex if the wrong level of abstraction is chosen

Visual Languages: Scratch, Storytelling Alice

TECHNISCHE UNIVERSITÄT DARMSTADT



Visual Language: Unreal Blueprint

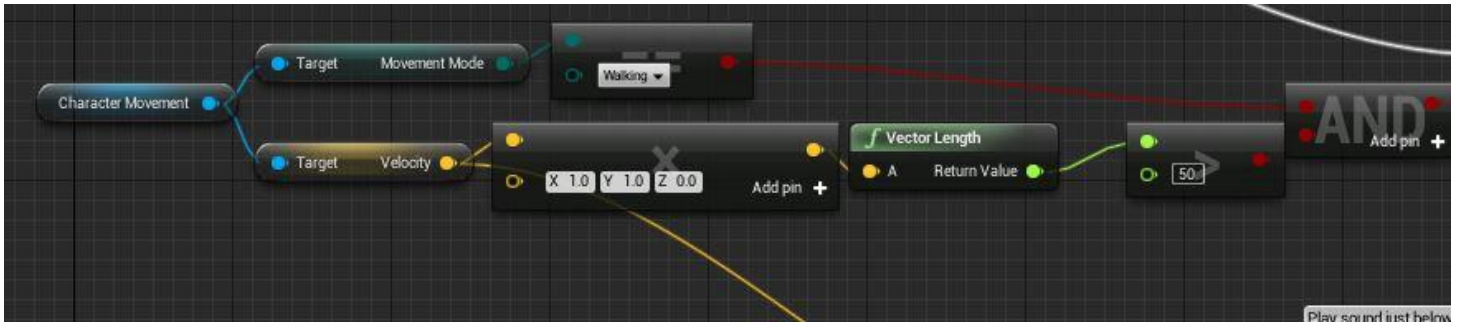
TECHNISCHE UNIVERSITÄT DARMSTADT

Added in Unreal Engine 4

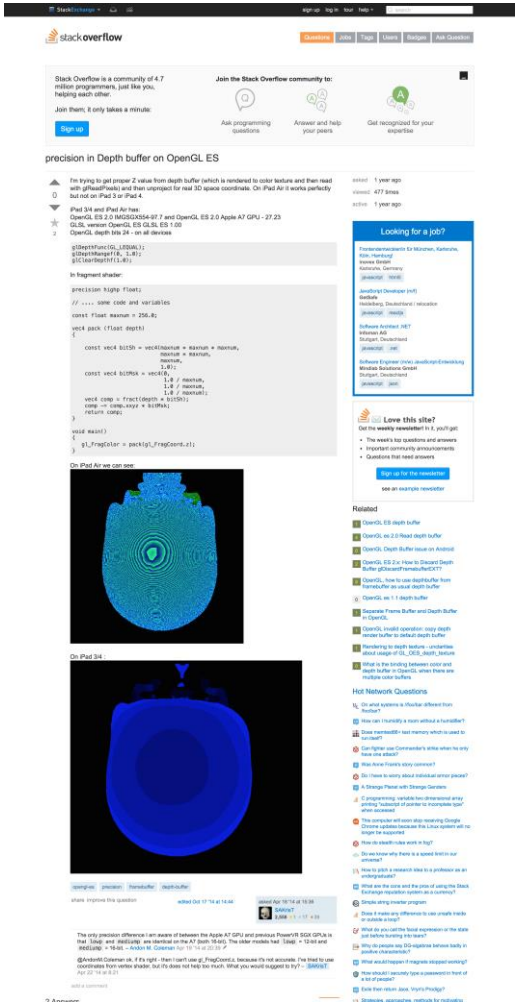
Can modify almost everything in the game

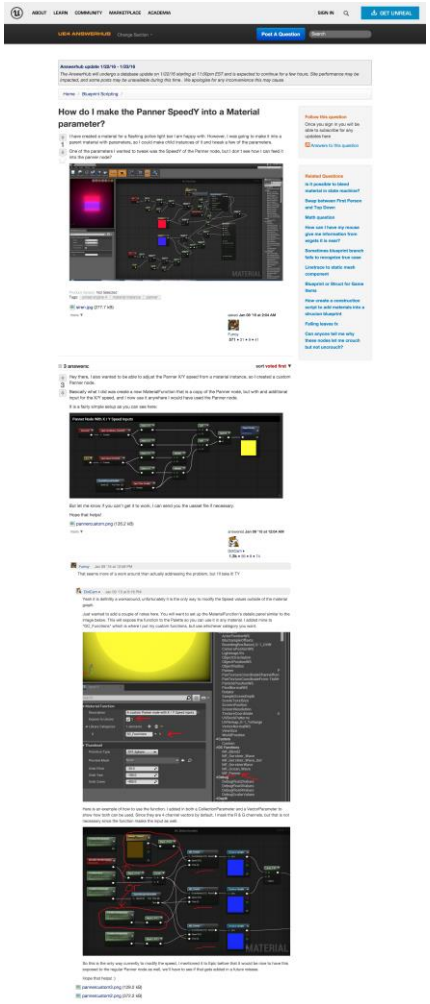
Can extend C++ classes

Graph-based scripting language



Game dev forum comparison





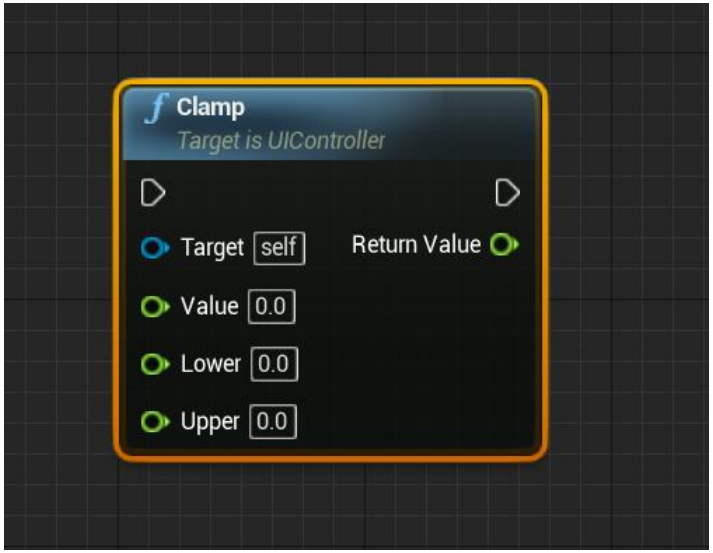
Unreal Blueprints

TECHNISCHE UNIVERSITÄT DARMSTADT

Can interact with C++ code

Exposed via preprocessor magic

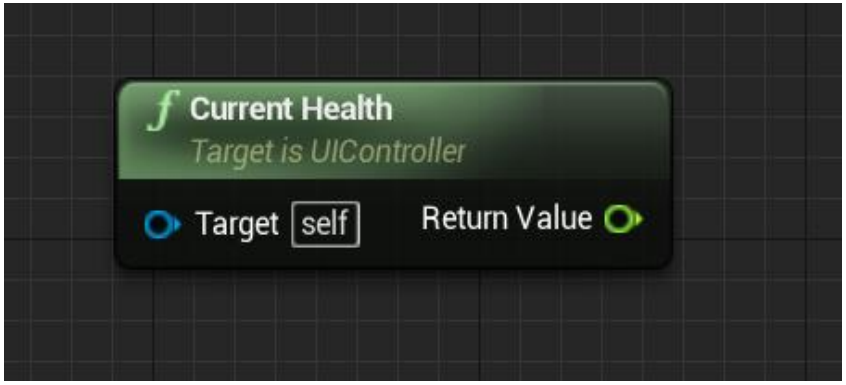
```
UFUNCTION(BlueprintCallable, Category = "GameTech")
float Clamp(float Value, float Lower, float Upper);
```



Unreal Blueprints

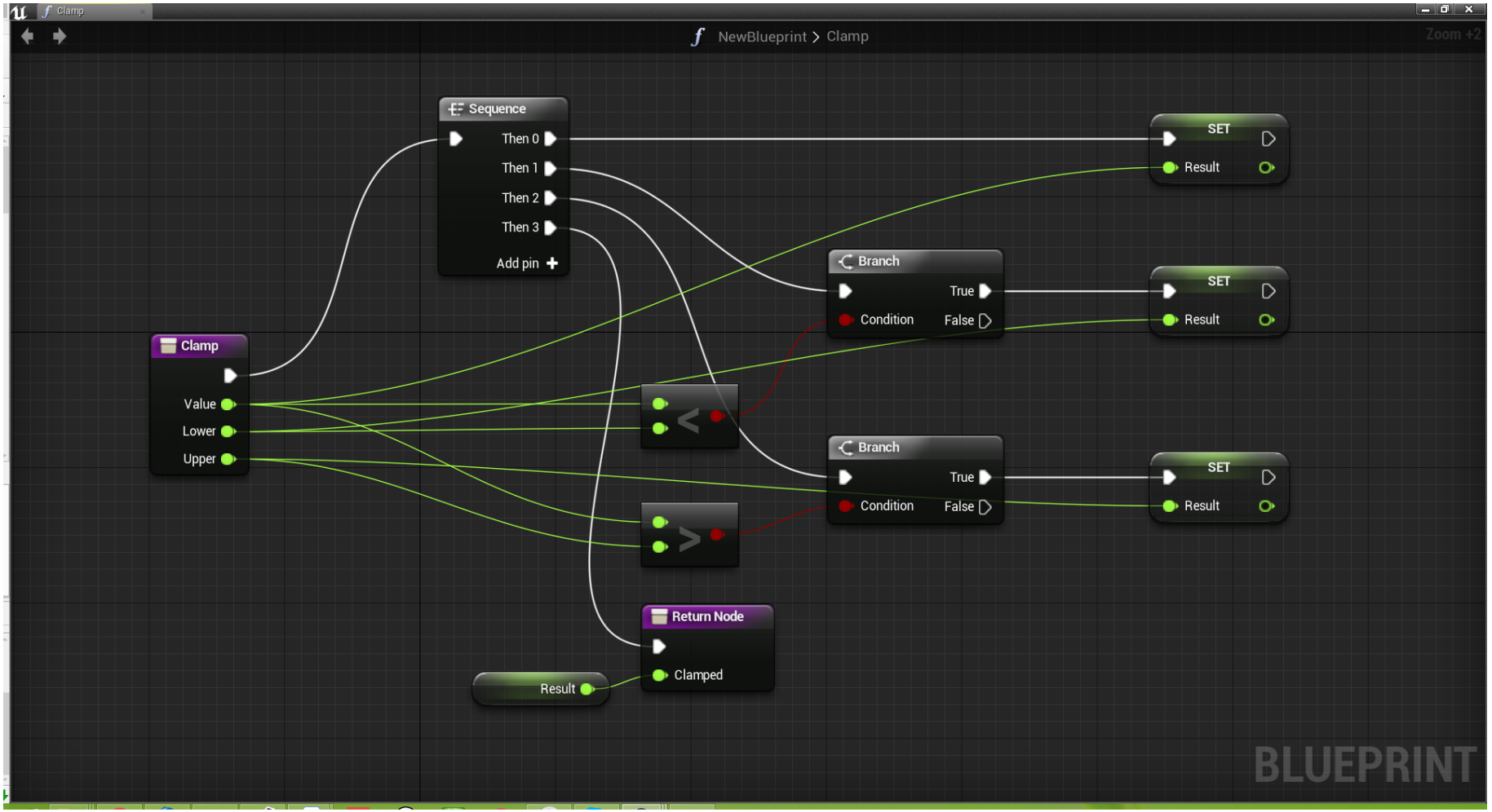
TECHNISCHE UNIVERSITÄT DARMSTADT

```
UFUNCTION(BlueprintCallable, Category = "GameTech")
float CurrentHealth() const;
```



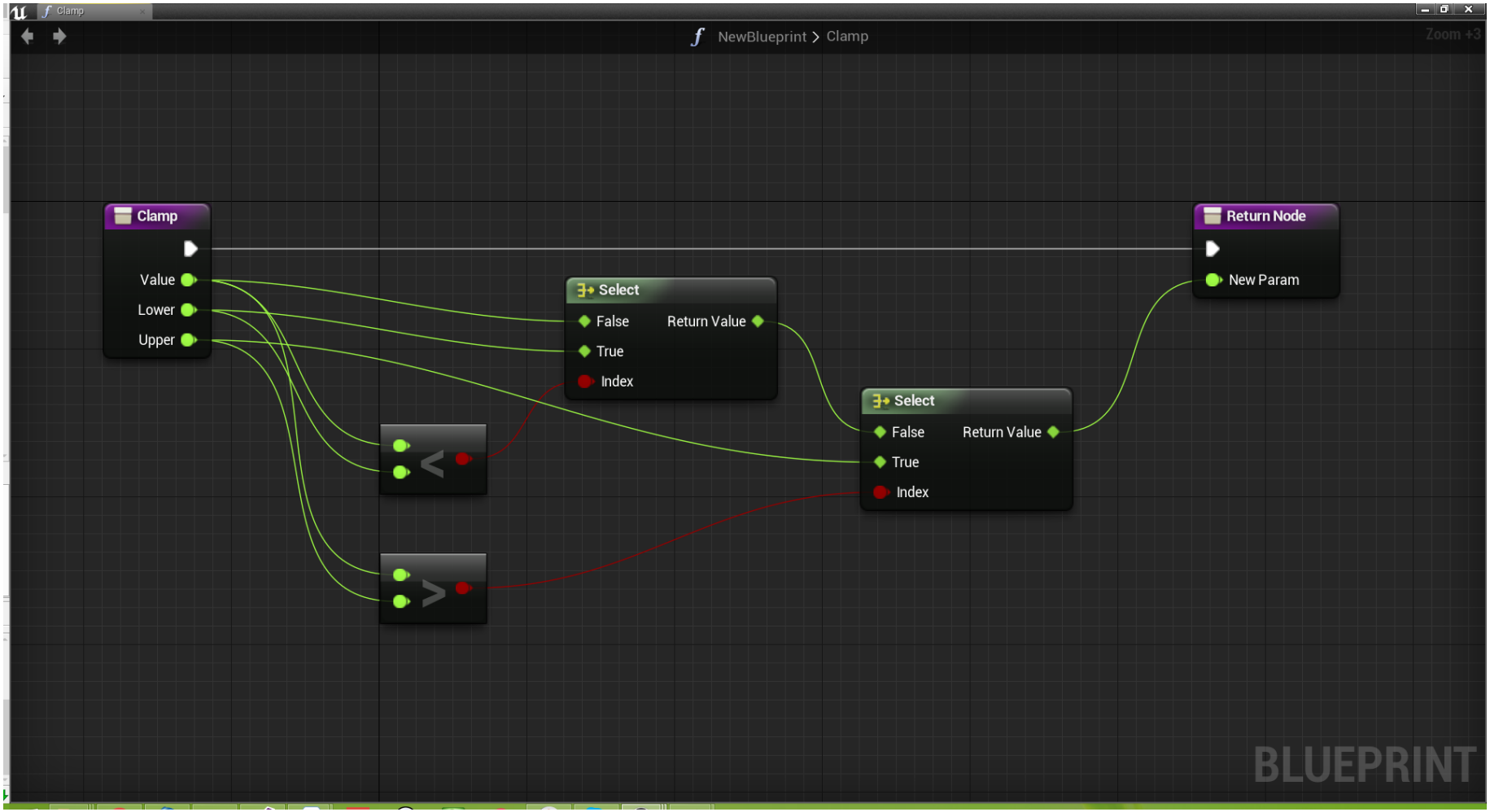
Programming in vs. into the language

TECHNISCHE UNIVERSITÄT DARMSTADT



Programming in vs. into the language

TECHNISCHE UNIVERSITÄT DARMSTADT



Visual Shader Programming

TECHNISCHE UNIVERSITÄT DARMSTADT

Combine simple operations

Functions as complex nodes

Advantages

- Debugging: Can show intermediate steps
- Duality Numbers/Colors
- Easy re-use of smaller graph elements

Shader Programming Example

TECHNISCHE UNIVERSITÄT DARMSTADT

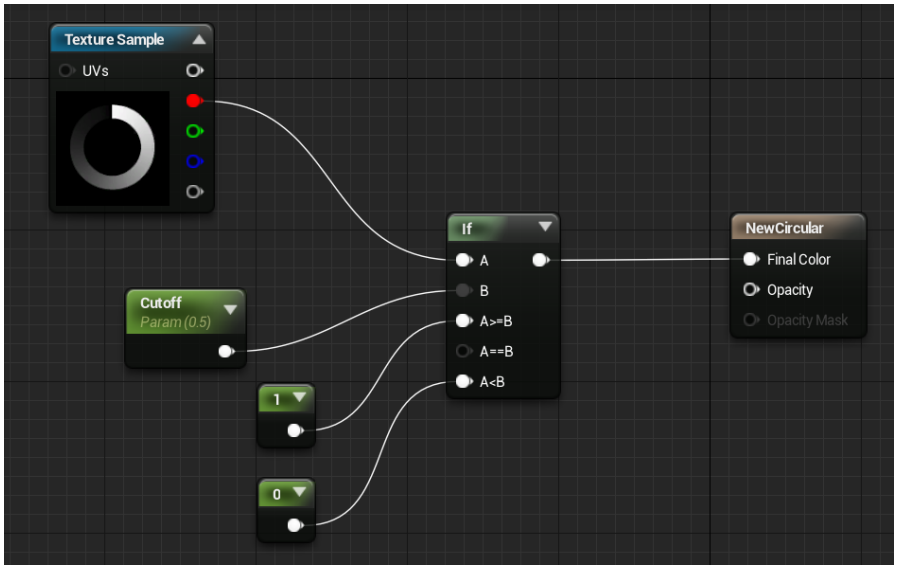
Creating a rounded progress bar

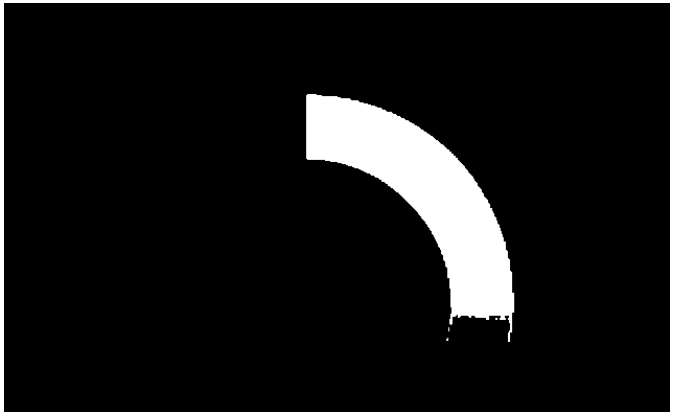
Usual approach

- Export a rounded gradient
- Cut off at a specified alpha between 0 and 1
- Problem: Can have floating point inaccuracies due to compression

Alternative: Create the required gradients in a shader

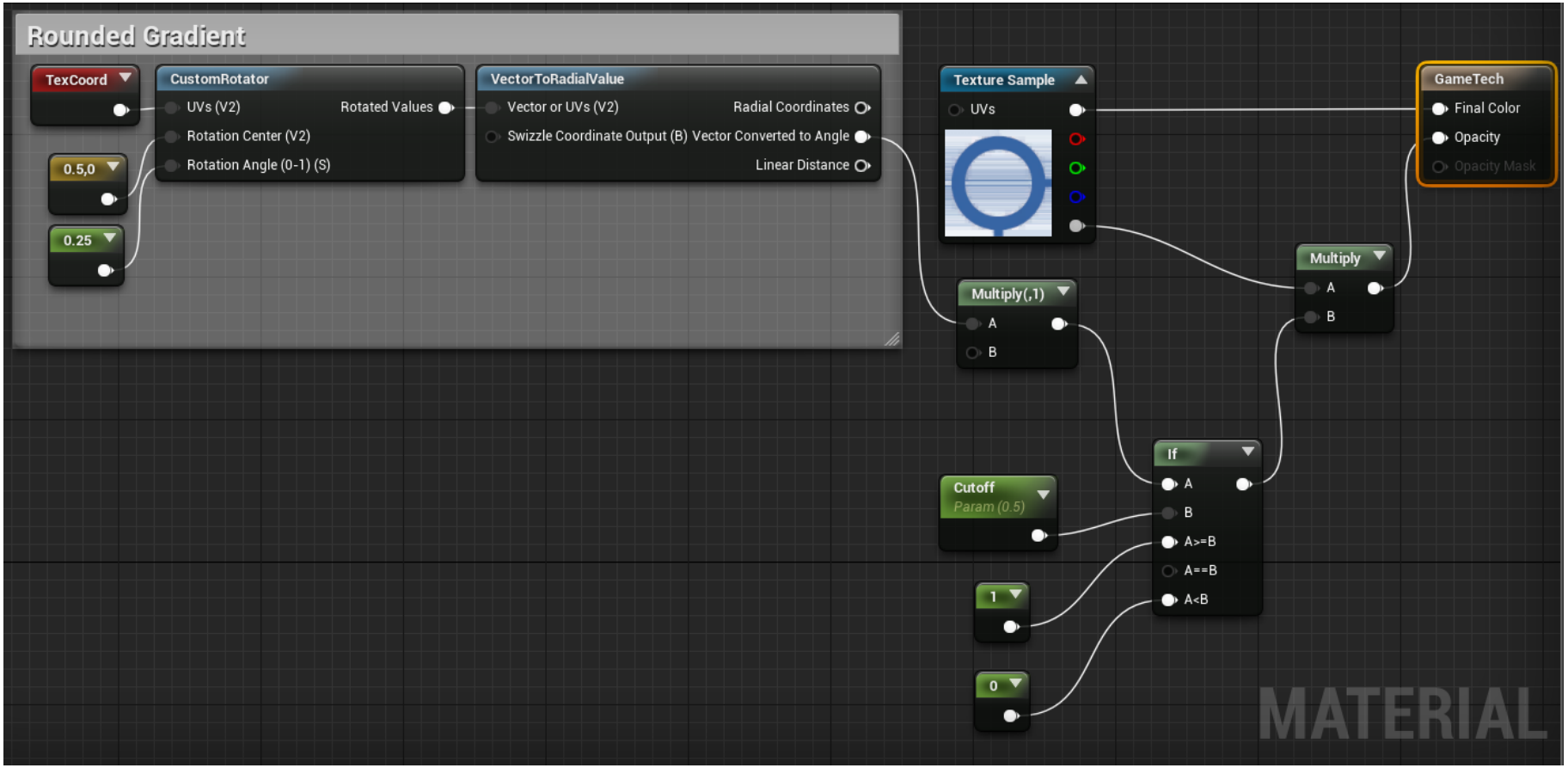
Gradient supplied as an image





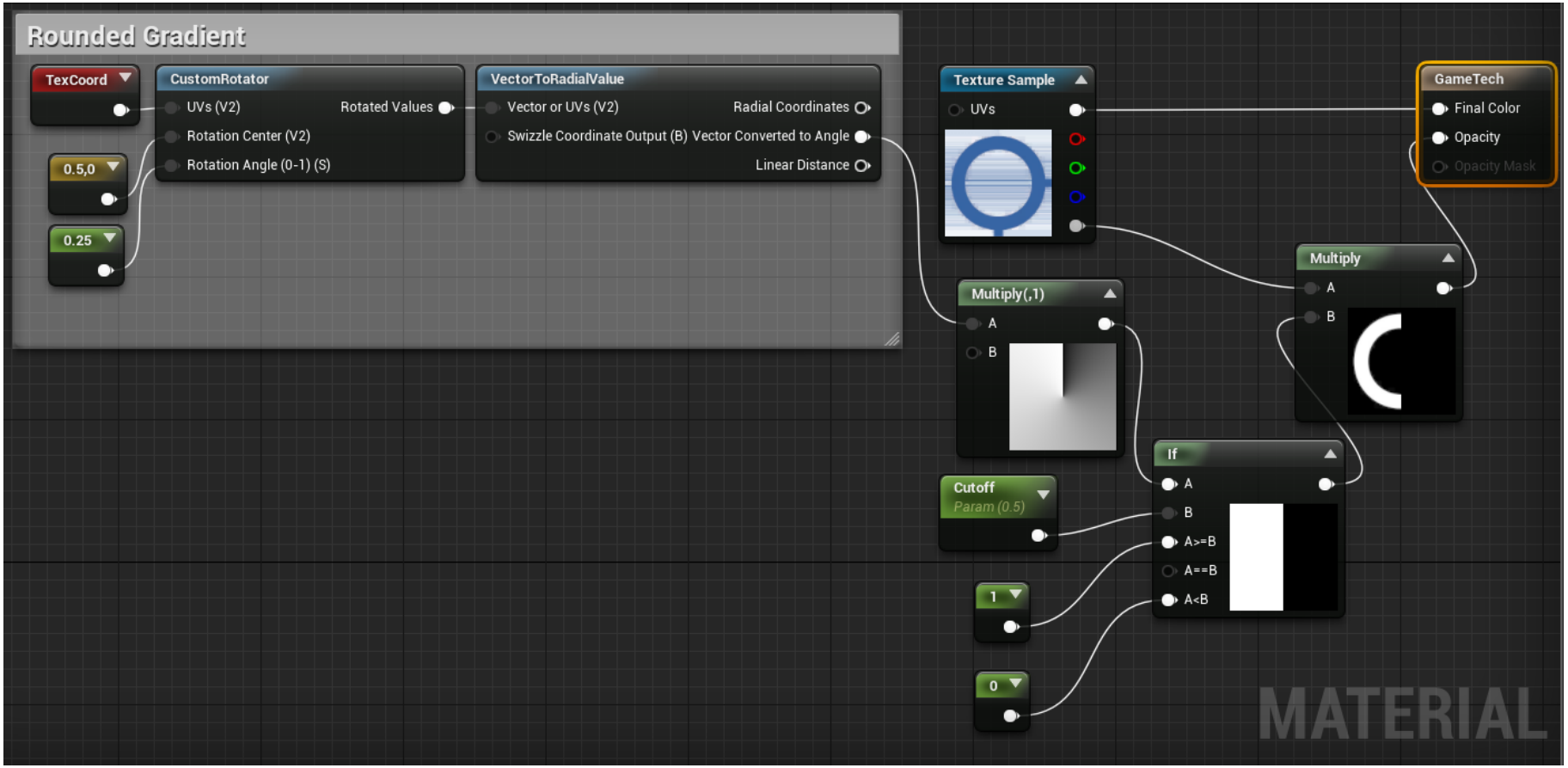
Gradient created in the shader

TECHNISCHE UNIVERSITÄT DARMSTADT



Visual debugging

TECHNISCHE UNIVERSITÄT DARMSTADT



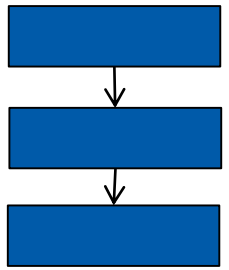
Result

Graph types for visual scripting languages

TECHNISCHE UNIVERSITÄT DARMSTADT

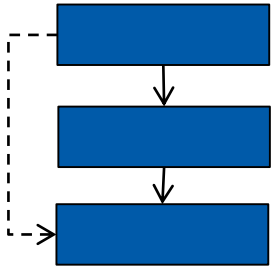
Single branch tree

- Analogue to function without conditions or jumps
- Easiest to implement, but very inflexible



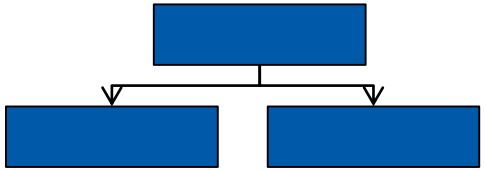
Single branch tree with jumps

- Uses in first version of Unity Adventure Creator
  - …probably out of necessity



Trees

- Allow conditions to be visualized effectively

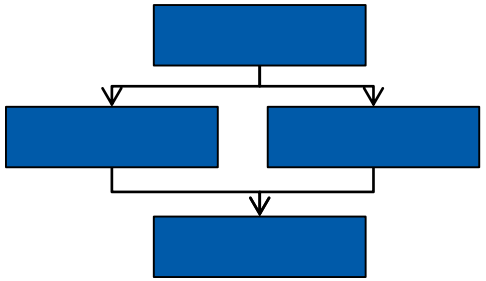


Graph types for visual scripting languages

TECHNISCHE UNIVERSITÄT DARMSTADT

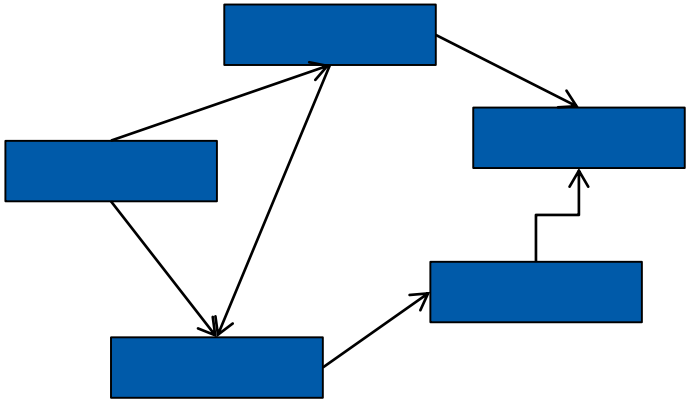
Series-Parallel Digraphs

- Source and Drain similar to electrical circuit
- Always ends in a node
- Advantage over tree: Can be be layouted well automatically, works for scripts with branches and a common end



Graph

- No restrictions
- Most often freely drawn by designer
- Hard to layout automatically
- → Most common for visual scripting languages in games



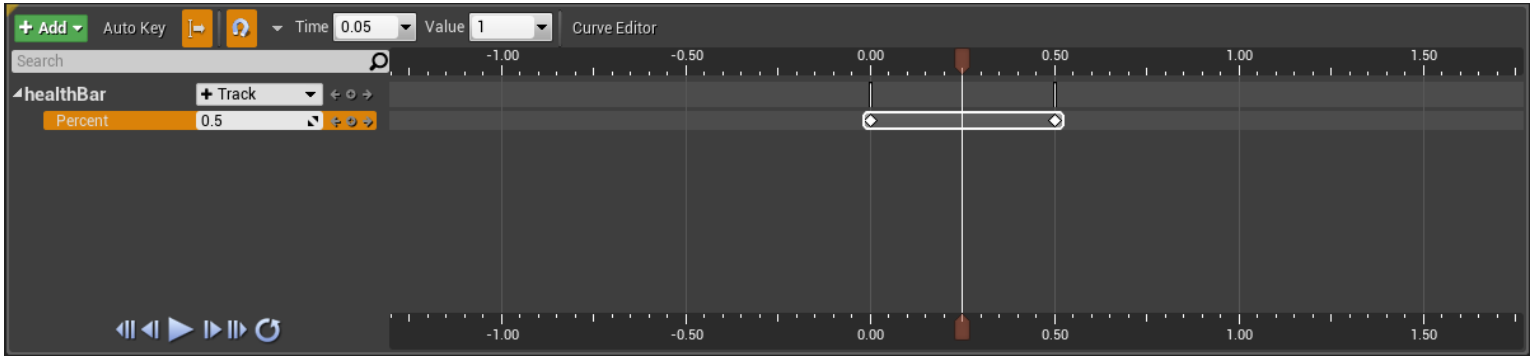
Timelines

TECHNISCHE UNIVERSITÄT DARMSTADT

For specifying choreographed events

Scrubbing

Can be mixed, e.g. Timelines in UMG (Blueprint Unreal Engine 4)

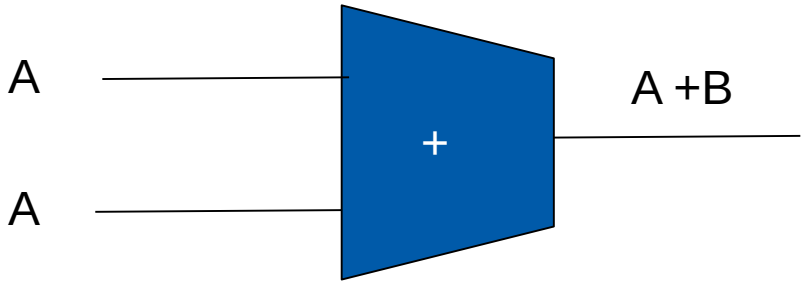


Data in visual scripting languages

TECHNISCHE UNIVERSITÄT DARMSTADT

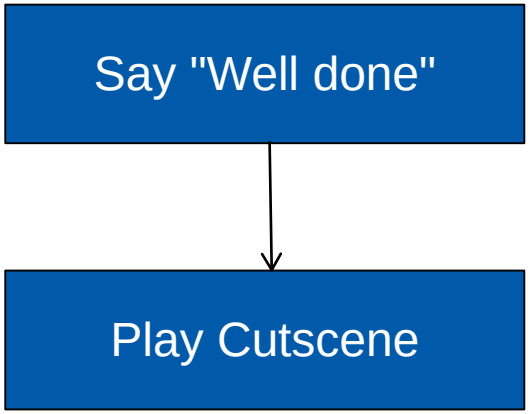
Passed around implicitly

- Arguments to individual actions
- Look up, e.g. from a blackboard architecture



Passed around explicitly in the graph

- Exit slots for output variables
- Input slots for input
- Advantage: Can create nodes to change input



Domain-Specificity

TECHNISCHE UNIVERSITÄT DARMSTADT

If we restrict the domain of our language, we gain specialization and convenience at the cost of generality

One extreme example: Unreal Blueprints

- Replacement for general-purpose programming language
- Operate on numbers, strings, …
- Define functions, use if/else, switch, …

Example from Kha/Haxe-based VR adventure

TECHNISCHE UNIVERSITÄT DARMSTADT

Domain-Specificity

TECHNISCHE UNIVERSITÄT DARMSTADT

```
game.AddInventoryItem("Book");
game.DrawSubImage2("examine_bookshelf", "atlas", 1173,
      579, 366, 241, 2082, 1154, 366, 241);
game.PlaySound("add_inventory");
game.DisableHotspot("Instructions");
```

Advantages

- Can provide specific commands, e.g. Character.Say → play sound, show dialogue, …
- Easier for non-programmers

Note: We can build from general-purpose languages

Architectures for Scripting

TECHNISCHE UNIVERSITÄT DARMSTADT

Scripted Callbacks

- Most of the behaviour is hard-coded
- Code calls hook functions that are implemented in scripting language

Scripted Event Handlers

- Special case of callbacks
- Allows game objects to react to certain types of events

Extending game object types/define new ones

- Via inheritance or composition/aggregation
- E.g. UnrealScript

Architectures for Scripting

TECHNISCHE UNIVERSITÄT DARMSTADT

Scripted Components or properties

- In component-based game engine architectures
- Define the component by the scripting language
- Used in Dungeon Siege (2002)
- Used by Unity

Script-driven engine systems

- Whole sub-system created in scripting language
- E.g. game object model in script
- Only calls hard-coded parts when needed (e.g. performance-critical parts)

Script-driven games

- Mainly script, game engine more of a library
- E.g. Panda3D

Game Engine Integration

TECHNISCHE UNIVERSITÄT DARMSTADT

Embed the virtual machine (often written in C or C++)

Interface to/from native code

- Functional language
  - Look up the function‘s byte code and run it, providing arguments
- Object-oriented language
  - Create/destroy instances, call member functions
- Two-way communication
  - Allow script functions to call native code
  - Often realized by registering functions with the scripting language
  - Can be automized if the native language supports RTTI (e.g. see Lua integration into C#)

Game Engine Integration

TECHNISCHE UNIVERSITÄT DARMSTADT

Referring to Game Objects

Numerical Handles

- Simple to use/set up
- Can be confusing

Strings with names

- Easier to use
- More memory used, string comparisons, miss-types names

Hashed string ids

- Reduce to integer for the engine

AI Scripting - Finite State Machines

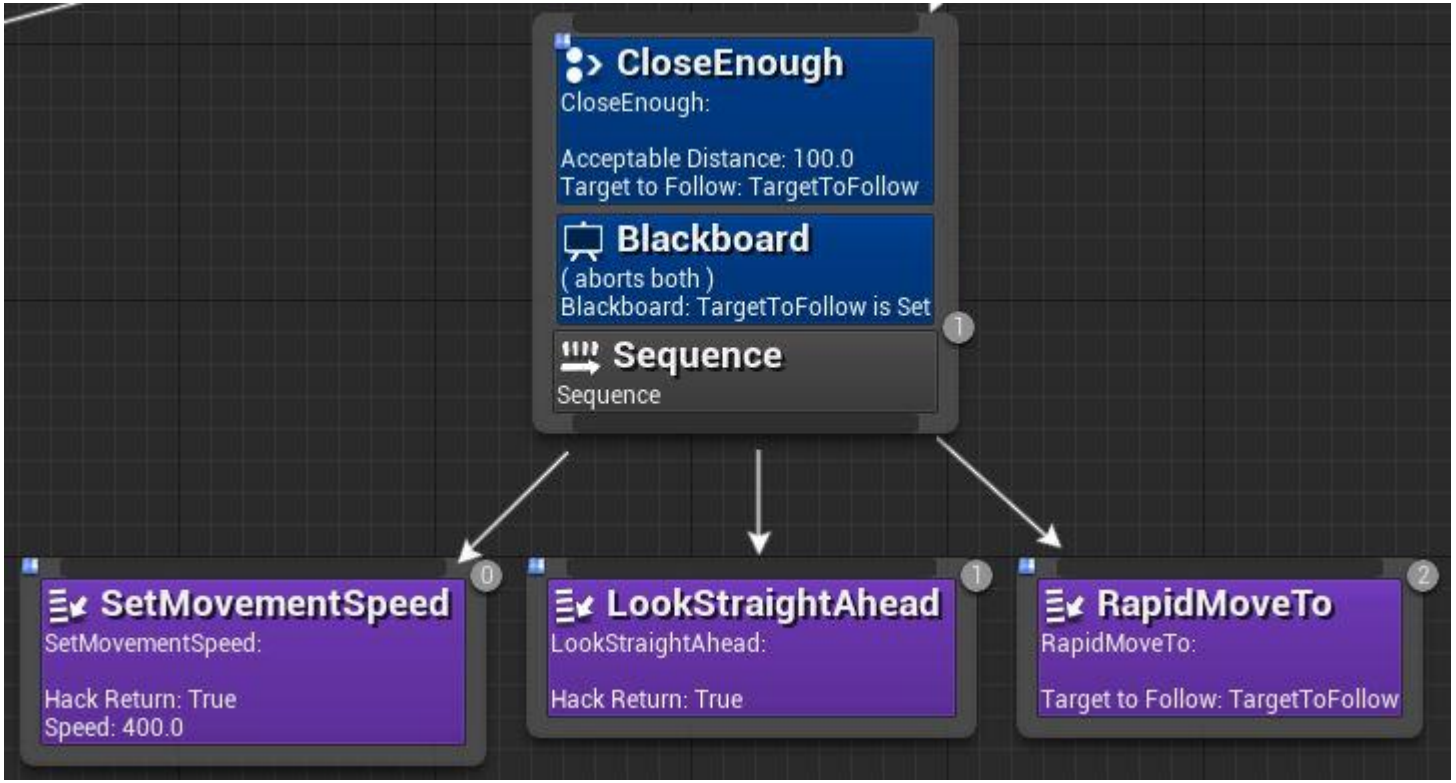
TECHNISCHE UNIVERSITÄT DARMSTADT

As seen in the AI lecture, FSM are often at the core of AI and game logic code

Specific support in the scripting language

- Custom syntax for states
- Mirrored in the game object model
- Example Uncharted Engine
  - Each script can have multiple states
  - Different event handlers, …

Unreal Behaviour Trees



https://docs.unrealengine.com/latest/INT/Engine/AI/BehaviorTrees/HowUE4BehaviorTreesDiffer/index.html

Multithreaded scripts

TECHNISCHE UNIVERSITÄT DARMSTADT

Usually done via cooperative multitasking

Scripts explicitly yield to other scripts

- Wait for x seconds
- Wait for x frames

Examples

- Can be realized in Lua
- Unity

Synchronized via signals

- E.g. WAIT_UNTIL(signal)

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

Non-programmer friendly

- Designers can test/iterate quickly
- Mod support

Quick iteration times

- More simple to program than full programming language
- Hot reload

But: Not for everything

- Performance critical code
- Complex code

Domain-specificity

C++ 11, C++14 (and Games)

TECHNISCHE UNIVERSITÄT DARMSTADT

ISO/IEC 14882:2011

Aka C++11

Many features used in current game engines

- E.g. UE4 uses many of these additions

Depends on your target compilers

Depends on your general library architecture

- E.g. if templates are used at all
- But some features a win in most situations, e.g. move semantics

shared_ptr

TECHNISCHE UNIVERSITÄT DARMSTADT

```
class Foo;

shared_ptr<Foo> myFoo = new Foo();
shared_ptr<Foo> anotherFoo = myFoo;

anotherFoo = myFoo = nullptr; // The object previously
      referenced is now deleted
```

Auto, decltype

TECHNISCHE UNIVERSITÄT DARMSTADT

```
int x = 5;
auto x = 5;

vector<int> vec;

// instead of vector<int>::iterator itr
auto itr = vec.iterator();

int x = 3;
decltype(x) y = x; // same thing as auto y = x;
```

Direct data initialization

TECHNISCHE UNIVERSITÄT DARMSTADT

```
class Foo {
 public: int x = 5;
 int y = 4;
 int arrray[5] {1,2,3,4};
}
```

Super convenient

Lambda expressions

TECHNISCHE UNIVERSITÄT DARMSTADT

[*capture-list* **] (** *params* **) -> ** *ret* **{** *body* **}**

```
[ ] (int x) -> bool { return x > 5; }

int compare = 5;
[ ] (int x) -> bool { return x > compare; } // Will not
            work, need to capture "compare"

int compare = 5; [compare] (int x) -> bool { return x >
      compare; } // OK
```

Use for timers, search function predicates, …

Simple to declare function pointer

constexpr


```
constexpr int sqr(int arg) {
  return arg*arg;
}
```

Allows the compiler to compute the result at compile time

```
int array[sqr(5)]; //Works, allows the compiler to
  compute array size during compilation
```

Not supported in Visual Studio 2013 ☹

Static_assert

TECHNISCHE UNIVERSITÄT DARMSTADT

```
static_assert ( bool_constexpr , message );
```

Asserts properties of the code during compilation

Boolean parameter has to be a constexpr for obvious reasons

Range-based for loops

TECHNISCHE UNIVERSITÄT DARMSTADT

```
TArray<AActor*> Actors;

for (int32 ActorIndex=0; ActorIndex<Actors.Num();
 ActorsIndex++) {
 AActor* Actor = Actors[ActorIndex];
 Actor->SetActorLocation(NewLocation);
}


for (AActor* Actor : Actors) {
 Actor->SetActorLocation(NewLocation);
}
```

Move semantics

TECHNISCHE UNIVERSITÄT DARMSTADT

x+(y*z);
(y * z) is an rvalue
→ It can only be on the right side of =

```
A = (y * z); // Ok
(y * z) = A; // Not ok

MemoryPage(MemoryPage&& other): size(0), buf(nullptr) {
  // pilfer other's resource
  size=other.size;
  buf=other.buf;
  // reset other
  other.size=0;
  other.buf=nullptr;
}
```

→ We know that a MemoryPage&& reference will not be used further
→ No need to copy the object first

Using

TECHNISCHE UNIVERSITÄT DARMSTADT

Like a typedef

But also compatible with templates

```
template <typename T>
using my_type = whatever<T>;
my_type<int> variable;

template <typename U> struct baz {
 my_type<U> _var_member;
}
```

C++14

Not as widely adopted as C++11 yet

http://www.drdobbs.com/cpp/the-c14-standard-what-you-need-to-know/240169034

Improvement of C++11 features

E.g. auto for return types

Generic lambdas

Deprecated attribute

Binary literals

Thank you!


thank you
danke
рахмат
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
mersi
barka
welalin
tack
spas
dank je
gracias
tapadh leat
faafetai lava
vinaka
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
kiitos
dankie
mauruuru
хвала
asante
manana
obrigada
tenki
nanni
nandri
dhanyavad
bedankt
hvala
köszönöm
akun
dankon
djiere dieuf
tau
mochchakkeram
enkosi
dziękuję
дякую
mamnun
murakoze
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
go raibh maith agat
sobodi
dėkuji
obrigado
sagolun
sukriya
kop khun krap
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
rahmet
shukriya
мерси
kam sah hamnida
rahmat
terima kasih
diolch
dhanyavadagalu
감사합니다
xièxie
ευχαριστώ
merci