Project Nyx

NyxCore

Moddertreffen 2020

Phoenix Tales

[Inhalt]

1. ProjectNyx
    a. Projektidee
    b. Design-Philosophie
2. NyxCore
    a. Konzept
    b. Charaktersystem
    c. Daedalus-Erweiterung
    d. Ausblick
3. OpenSource

[ProjectNyx] Projektidee

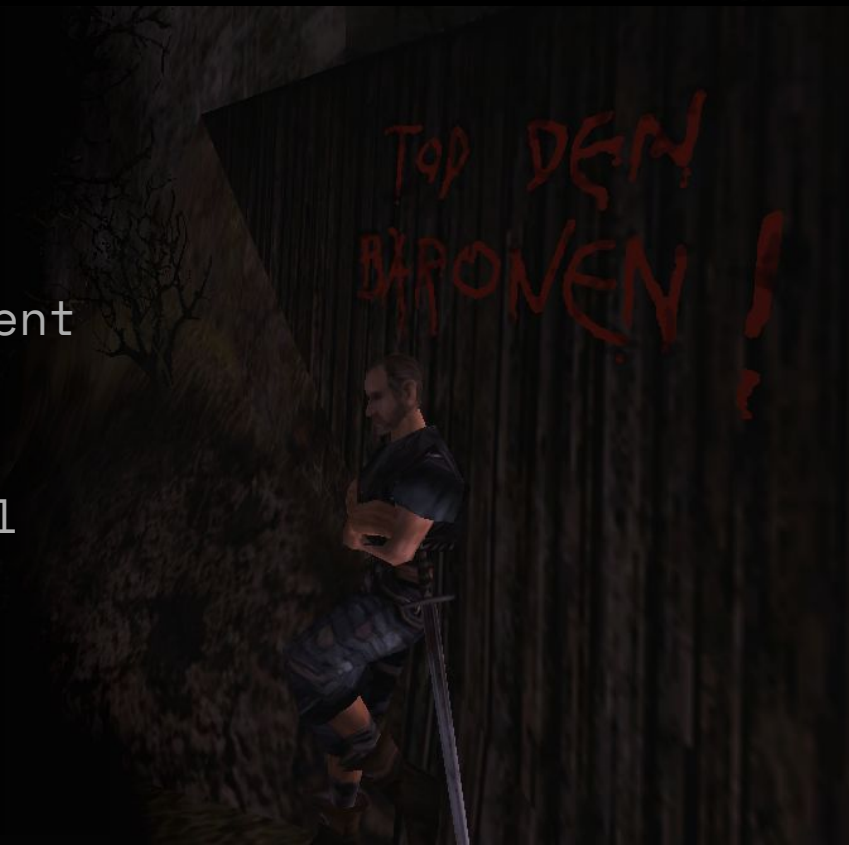
- Wiederauferstehung der ***Gothic Alpha*** im Hinblick auf Artstyle, Content und Gamedesign

- Release in ***zwei Akten***
  I. Orpheus (Alpha)
  II. Nemesis (Sequel)

- ***Gesamterlebnis*** aus Spiel, Art und Comic

[ProjectNyx] Design-Philosophie

- ***Radikale Visualisierung***
  keine EXPs, keine Skill-Trees
- ***Atmosphärische Dichte***
  Welt als Bühne, kein Füll-Content
- ***Erzählerische Tiefe***
  eine Story, mehrere Blickwinkel
- ***Ästhetik des Verfalls***
  Wabi-Sabi, Rost und Patina



[NyxCore] Konzept

[NyxCore] Konzept

- ***Kernfunktionalität für immersives Gameplay***
  erweitertes Charaktersystem, minimalistische Menüs
- ***Erweiterung der Gothic-Engine mittels DLLs***
  Implementierung in C++ im Union-Framework
- ***Erweitertes Daedalus-Scripting***
  erweiterter zParser, beliebige Engine-Externals
- ***Modulares Design, Abwärtskompatibilität***
  konfigurierbares Plugin, keine Änderung der Gothic.exe

[NyxCore] Charaktersystem

[NyxCore] Charaktersystem

- ***Kraft, Geschicklichkeit, Konstitution, Willenskraft***
  skalieren die Fähigkeiten jedes NPCs

- ***30+ neue Talente mit eigener Progression***
  Meucheln, Items werfen, Türen zertrümmern...

- ***Survival-Aspekte***
  Ausdauer, Vergiftung, Wahnsinn, Inventarbegrenzung

- ***Psioniker-Magie***
  Zaubern durch Willenskraft ohne Runen

[NyxCore] Charaktersystem

```
class oCNpcEx : public oCNpc {
 public:
     zCLASS_UNION_DECLARATION(oCNpcEx);

     TNpcAttributes  attr;
     TNpcExperience exp;
     TNpcSkills skills;

     oCNpcEx();

     virtual void ProcessNpc();

     virtual void Archive(zCArchiver& ar);
     virtual void Unarchive(zCArchiver& ar);
 };
```

Der erweiterte NPC erbt von Gothics NPC-Klasse und verhält sich damit genauso wie alle NPCs bisher.

[NyxCore] Charaktersystem

```
class oCNpcEx : public oCNpc {
 public:
     zCLASS_UNION_DECLARATION(oCNpcEx);

     TNpcAttributes  attr;
     TNpcExperience exp;
     TNpcSkills skills;

     oCNpcEx();

     virtual void ProcessNpc();

     virtual void Archive(zCArchiver& ar);
     virtual void Unarchive(zCArchiver& ar);
 };
```

Er hat aber auch einige erweiterte Datenfelder. Hier z.B. neue Attribute, Erfahrungspunkte und Skills.

[NyxCore] Charaktersystem


```
class oCNpcEx : public oCNpc {
 public:
     zCLASS_UNION_DECLARATION(oCNpcEx);

     TNpcAttributes  attr;
     TNpcExperience exp;
     TNpcSkills skills;

     oCNpcEx();

     virtual void ProcessNpc();

     virtual void Archive(zCArchiver& ar);
     virtual void Unarchive(zCArchiver& ar);
 };
```

Hier überschreiben wir, was mit dem NPC in jedem Zyklus des Game-Loops geschieht, z.B. Regeneration oder Buffs.

[NyxCore] Charaktersystem

```
class oCNpcEx : public oCNpc {
 public:
     zCLASS_UNION_DECLARATION(oCNpcEx);

     TNpcAttributes  attr;
     TNpcExperience exp;
     TNpcSkills skills;

     oCNpcEx();

     virtual void ProcessNpc();

     virtual void Archive(zCArchiver& ar);
     virtual void Unarchive(zCArchiver& ar);
 };
```

Hier legen wir fest, wie der Zustand des NPC in den Spielstand geschrieben und wieder daraus geladen wird.

[NyxCore] Charaktersystem


```
class oCObjectFactoryEx : public oCObjectFactory {
 public:
     zCLASS_UNION_DECLARATION(oCObjectFactoryEx);
     virtual oCNpc* CreateNpc(int index);
 };


 oCNpc* oCObjectFactoryEx::CreateNpc(int index) {
     oCNpc* npc = new oCNpcEx();
     if (index != zPAR_INDEX_UNDEF)
         npc→InitByScript(index, 0);
     return npc;
 }


void OnInitFactory() {
     zfactory = new oCObjectFactoryEx();
 }
```

Die ObjectFactory erzeugt alle Objekte zur Laufzeit des Spiels z.B. NPCs und Items.

[NyxCore] Charaktersystem


```
class oCObjectFactoryEx : public oCObjectFactory {
 public:
     zCLASS_UNION_DECLARATION(oCObjectFactoryEx);
     virtual oCNpc* CreateNpc(int index);
 };
```

```
oCNpc* oCObjectFactoryEx::CreateNpc(int index) {
    oCNpc* npc = new oCNpcEx();
    if (index != zPAR_INDEX_UNDEF)
        npc->InitByScript(index, 0);
    return npc;
}
```

```
void OnInitFactory() {
     zfactory = new oCObjectFactoryEx();
 }
```

Unsere erweiterte Factory erzeugt statt normalen NPCs unsere NPCEx Instanzen. Diese können wie gewohnt per Daedalus-Skript beschrieben werden.

[NyxCore] Charaktersystem


```
class oCObjectFactoryEx : public oCObjectFactory {
 public:
     zCLASS_UNION_DECLARATION(oCObjectFactoryEx);
     virtual oCNpc* CreateNpc(int index);
 };


 oCNpc* oCObjectFactoryEx::CreateNpc(int index) {
     oCNpc* npc = new oCNpcEx();
     if (index != zPAR_INDEX_UNDEF)
         npc->InitByScript(index, 0);
     return npc;
 }
```

```
void OnInitFactory() {
     zfactory = new oCObjectFactoryEx();
 }
```

Hier sagen wir der Engine, dass sie unsere erweiterte ObjectFactory verwenden soll, wenn unser Plugin aktiv ist.

[NyxCore] Charaktersystem

[NyxCore] Daedalus-Erweiterung

```
int Npc_ChangePrimaryAttribute() {

  zCParser* par = zCParser::GetParser();
  int value;
  par→GetParameter(value);
  zSTRING attr_name;
  par→GetParameter(attr_name);
  oCNpcEx* npc_ex = (oCNpcEx*)par→GetInstance();

  if (attr_name == "ATR_STRENGTH") {
    //npc_ex→attr.Strength += value;
    npc_ex→attribute[NPC_ATR_STRENGTH] += value;
  }
  /* … */
  else if (attr_name == "ATR_WILLPOWER") {
    npc_ex→attr.Willpower += value;
  }

  return 0;
}
```

Hier holen wir die Parameter unserer Funktion (npc_ex, attr_name, value) vom zParser ab, welcher den Aufruf aus einem Daedalus-Skript liest.

[NyxCore] Daedalus-Erweiterung

```
int Npc_ChangePrimaryAttribute() {

  zCParser* par = zCParser::GetParser();
  int value;
  par→GetParameter(value);
  zSTRING attr_name;
  par→GetParameter(attr_name);
  oCNpcEx* npc_ex = (oCNpcEx*)par→GetInstance();

  if (attr_name == "ATR_STRENGTH") {
    //npc_ex→attr.Strength += value;
    npc_ex→attribute[NPC_ATR_STRENGTH] += value;
  }
  /* … */
  else if (attr_name == "ATR_WILLPOWER") {
    npc_ex→attr.Willpower += value;
  }

  return 0;
}
```

Hier verändern wir das gewünschte Attribut. Alte und neue Attribute bestehen nebeneinander und beeinflussen sich nicht gegenseitig.

[NyxCore] Daedalus-Erweiterung

```
parser→DefineExternal (
  "Npc_ChangePrimaryAttribute",
  Npc_ChangePrimaryAttribute,
  zPAR_TYPE_VOID,
  zPAR_TYPE_INSTANCE,
  zPAR_TYPE_STRING,
  zPAR_TYPE_INT,
  0
);
```

Name der Funktion in Daedalus.

[NyxCore] Daedalus-Erweiterung

```
parser→DefineExternal (
  "Npc_ChangePrimaryAttribute",
  Npc_ChangePrimaryAttribute,
  zPAR_TYPE_VOID,
  zPAR_TYPE_INSTANCE,
  zPAR_TYPE_STRING,
  zPAR_TYPE_INT,
  0
);
```

Aufgerufene Engine-Funktion.

[NyxCore] Daedalus-Erweiterung

```
parser→DefineExternal (
  "Npc_ChangePrimaryAttribute",
  Npc_ChangePrimaryAttribute,
  zPAR_TYPE_VOID,
  zPAR_TYPE_INSTANCE,
  zPAR_TYPE_STRING,
  zPAR_TYPE_INT,
  0
);
```

Daedalus Rückgabewert.

[NyxCore] Daedalus-Erweiterung

```
parser→DefineExternal (
  "Npc_ChangePrimaryAttribute",
  Npc_ChangePrimaryAttribute,
  zPAR_TYPE_VOID,
  zPAR_TYPE_INSTANCE,
  zPAR_TYPE_STRING,
  zPAR_TYPE_INT,
  0
);
```

Typen der Daedalus-Parameter.

[NyxCore] Daedalus-Erweiterung


```
/*****************************************************************************/
INSTANCE ItPo_Primary_Perm_Test(C_Item)
{
     /* … */
     description             = "(NYX) Trank der Attribute";
     TEXT[1]                 = "Bonus Stärke";                   COUNT[1]  = 10;
     TEXT[2]                 = "Bonus Geschicklichkeit";         COUNT[2]  = 10;
     TEXT[3]                 = "Bonus Konstitution";             COUNT[3]  = 10;
     TEXT[4]                 = "Bonus Willenskraft";             COUNT[4]  = 10;
     TEXT[5]                 = NAME_Value;                       COUNT[5]  = 1000;
};

     FUNC VOID UsePotPrimaryAttTest()
     {
          Npc_ChangePrimaryAttribute(self, "ATR_STRENGTH", 10);
          Npc_ChangePrimaryAttribute(self, "ATR_DEXTERITY", 10);
          Npc_ChangePrimaryAttribute(self, "ATR_CONSTITUTION", 10);
          Npc_ChangePrimaryAttribute(self, "ATR_WILLPOWER", 10);
     };
```

[NyxCore] Kleiner Rausschmeißer

[NyxCore] Ausblick

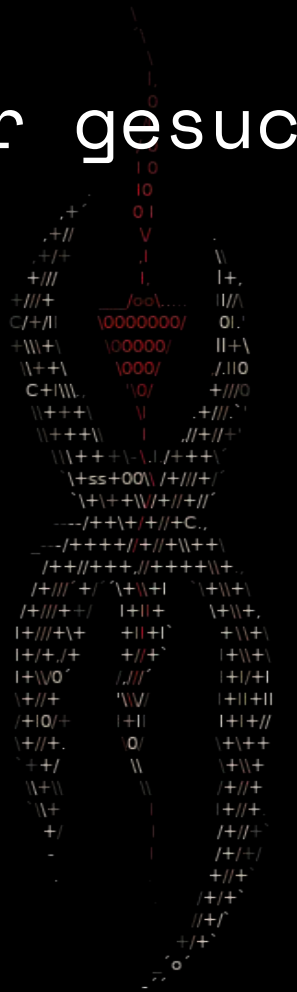
- ***Fertigstellung Charaktersystem***
  Psionik, Ausdauer, Gift
- ***Erweitertes Inventar***
  modulare Rüstungen, neue Item-Slots
- ***Immersive Menüs***
  Allround-Tagebuch, Crafting-HUD
- ***Erweiterte NPC-Interaktion***
  Goblins werfen, Finishing-Moves

[OpenSource] Mitstreiter gesucht!

- ***GitHub Repository***
  erster Open-Source-Release,
  sobald neues Attributssystem
  vollständig implementiert ist

- ***Engine-Programmierer***
  keine Modding-Vorkenntnisse
  nötig, aber Erfahrung mit C++

- ***weitere Stellen bei Project Nyx***
  Skripter, Grafiker, Comic-Artist



Project Nyx

No EXPs, no skill-trees, just roleplay.

(mail@)phoenixtales.de