Entwicklung eines Multiplattformprojektes für PC und Nintendo DS

Quo Vadis 2008

David.Salz@bitfield.de

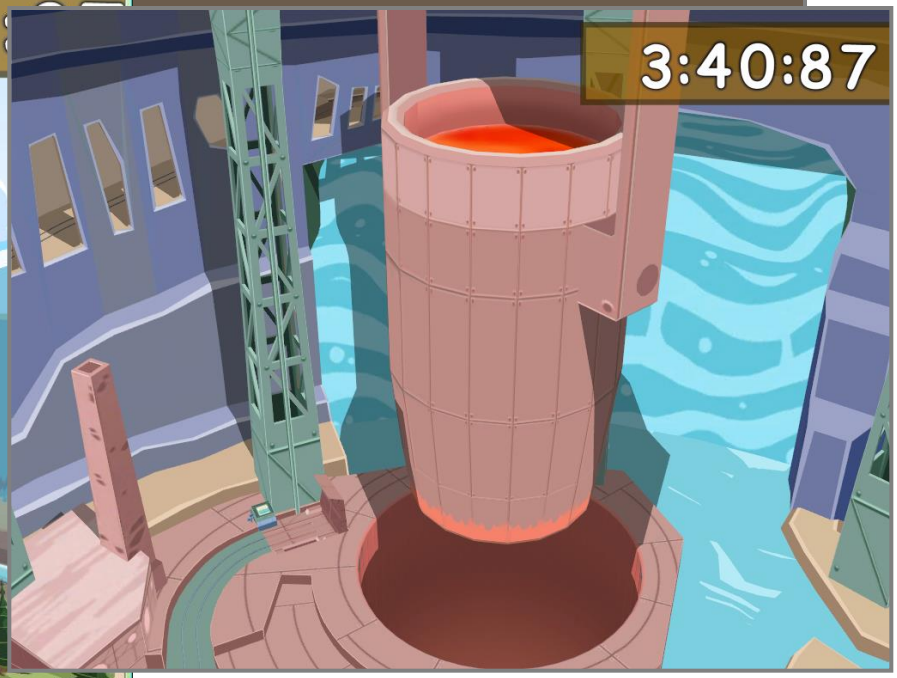
Daniel.Matzke@bitfield.de

BITFIELD

Über uns

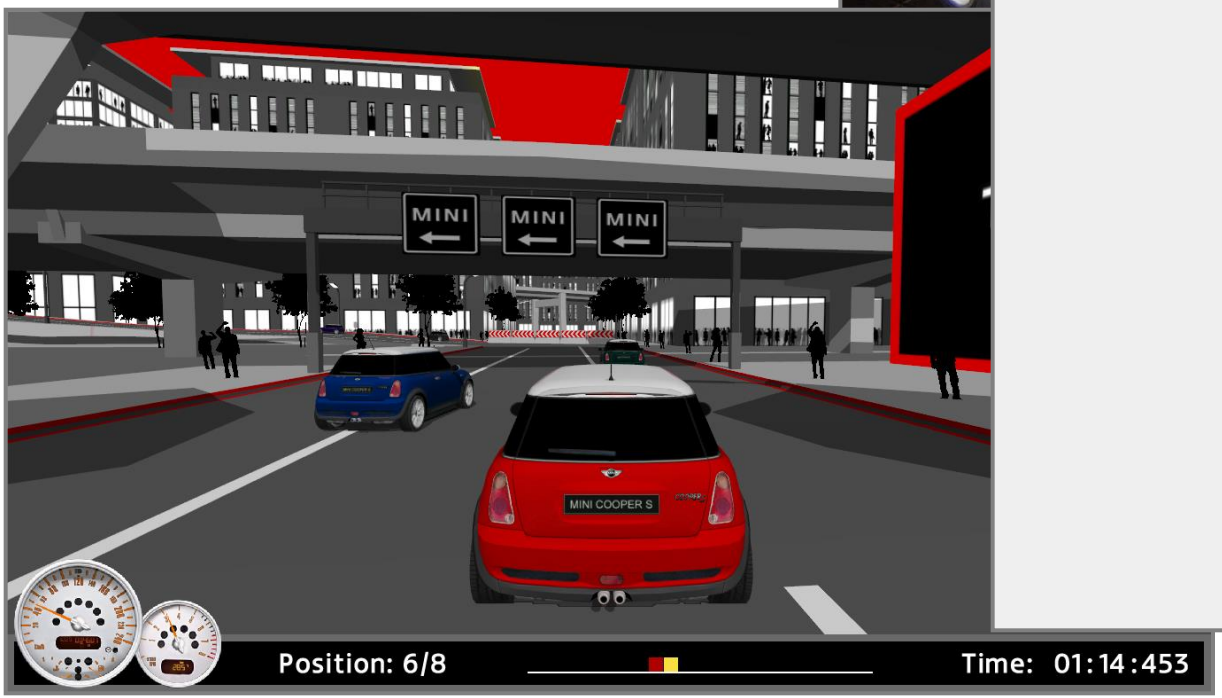
- gegründet Januar 2006
- Entwicklung für PC und Nintendo DS (DS-Entwickler seit 2006)
- Arbeitsgebiete:
  - Werbespiele für Ausstellungen & Messen
  - Casual Games für PC, DS
  - DS-Technologieprovider (BitEngineDS)
  - Produkt- und Architekturvisualisierung
- Team:
  - 2 Informatiker, 1 Architekt

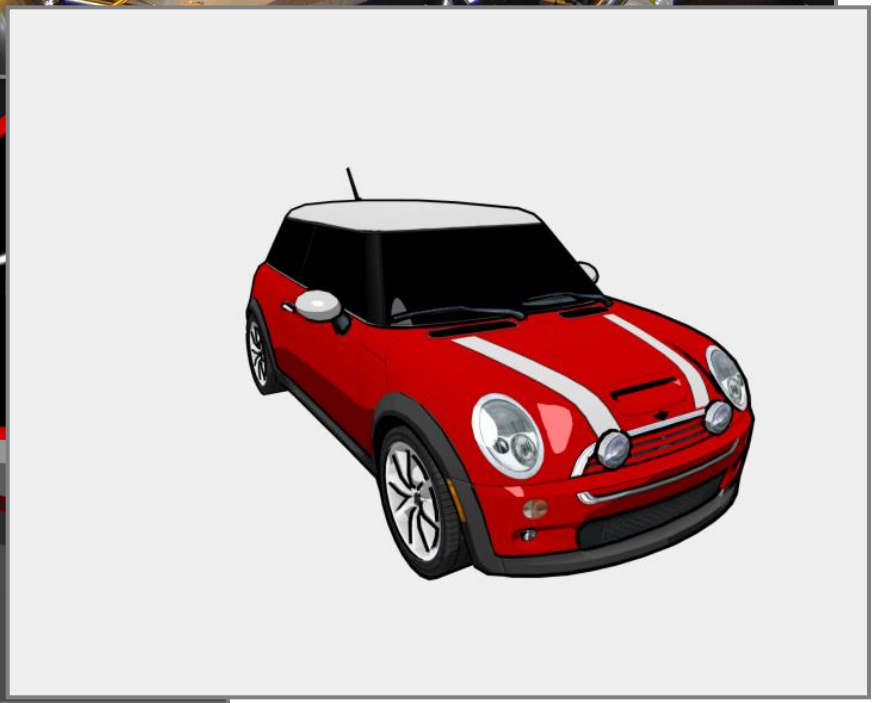
Railboy I/II



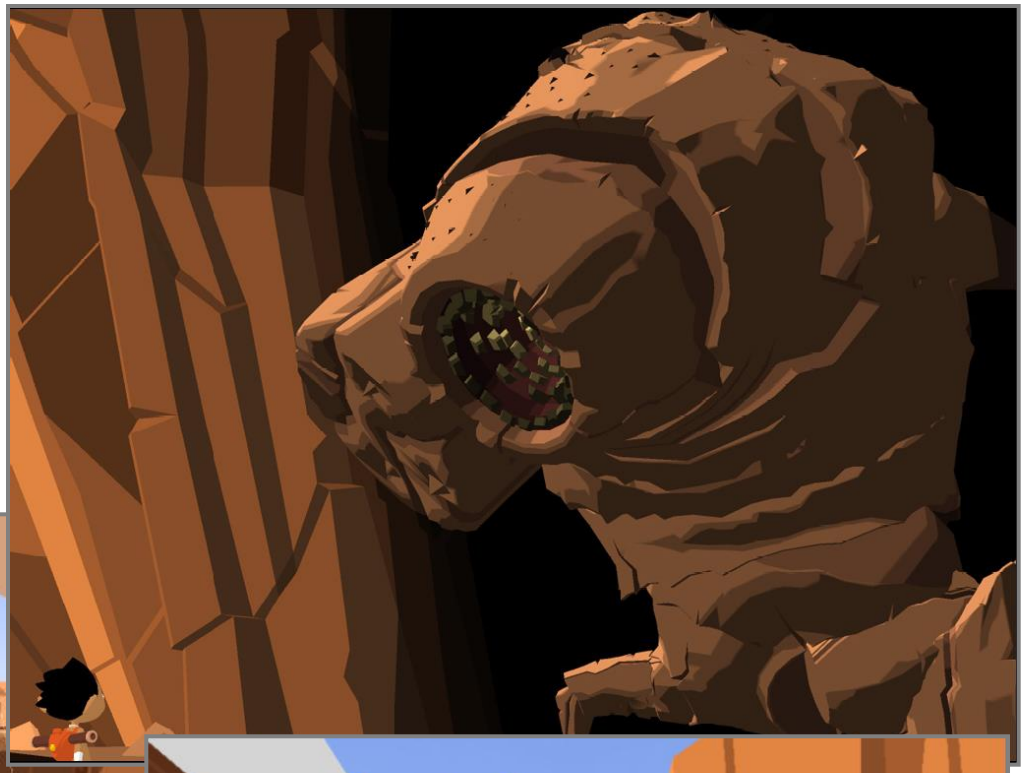
MINI-Racer

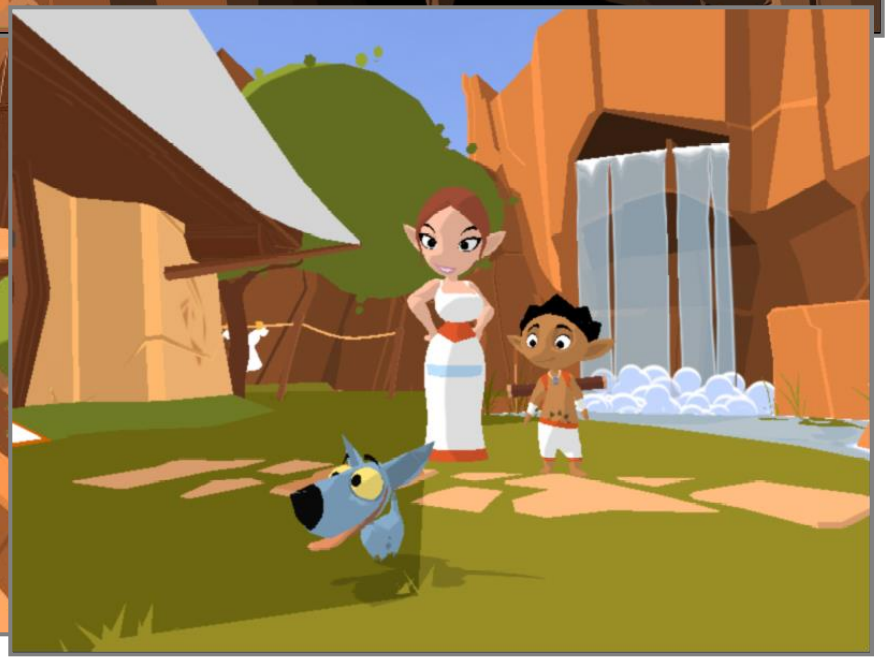

Push pedal to start the game


MINI
MINI
MINI
MINI COOPER S
Position: 6/8
Time: 01:14:453



“Didgeridoo“





Das Projekt

- Erstes DS-Projekt
  - Technik noch nicht vorhanden/nicht erprobt
  - neuer Workflow
    → sollte geringe Komplexität haben
- Vor- bzw. querfinanziert durch andere, parallele Projekte

Das Projekt

- Puzzlespiel, AT: „Puzzle Island“
- Klötzchen-Schieben nach dem Vorbild eines Brettspiels...
  - ~350 Puzzles
  - neue physikalische Spielsteine
  - Weltkarte
  - Story


ADVENTURES
ON
PUZZLE
ISLAND
ABENTEUER
FREIES SPIEL
SCHATZKAMMER

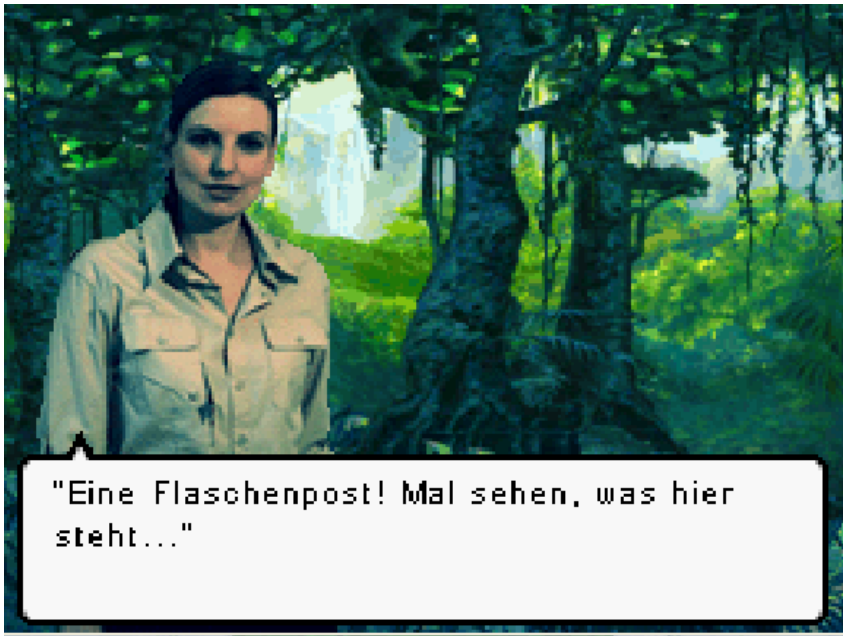

In den Weiten des Ozean kämpft ein Schiff allein mit den unerbittlichen Mächten der Natur. Doch der Kampf ist aussichtslos,


BERÜHRE HIER DEN BILDSCHIRM
ABBRECHEN


Und das ist der Ausgang. Sobald der Spielerstein dieses Feld erreicht, ist das Rätsel gelöst. Der Weg dahin, allerdings,
ZURUCK
VOR
MENU
BERÜHRE HIER DEN BILDSCHIRM


"Sehr gut! So schwer war das ja gar nicht."
ZURUCK
VOR
MENU
BERÜHRE HIER DEN BILDSCHIRM

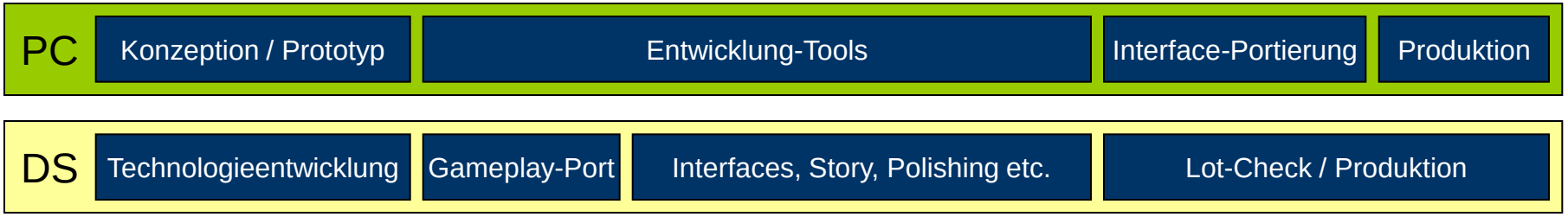

LEVEL
1-01
GOLD
3


"Eine Flaschenpost! Mal sehen, was hier steht..."


SPIELEN
MENÜ

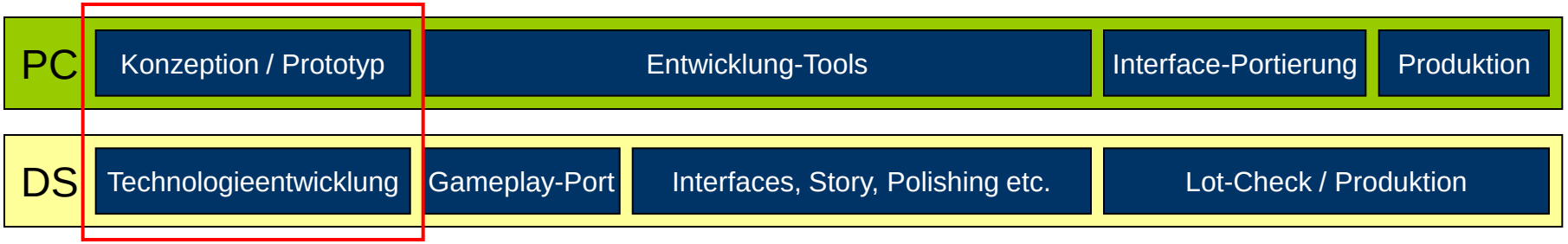

BERÜHRE HIER DEN BILDSCHIRM
ABBRECHEN

Übersicht - Projektphasen



- Team:
  - 1 Grafiker
  - 1 Designer (50%)
  - 2 Programmierer (50%)
- Dauer ca. 10 Monate

Phase 1

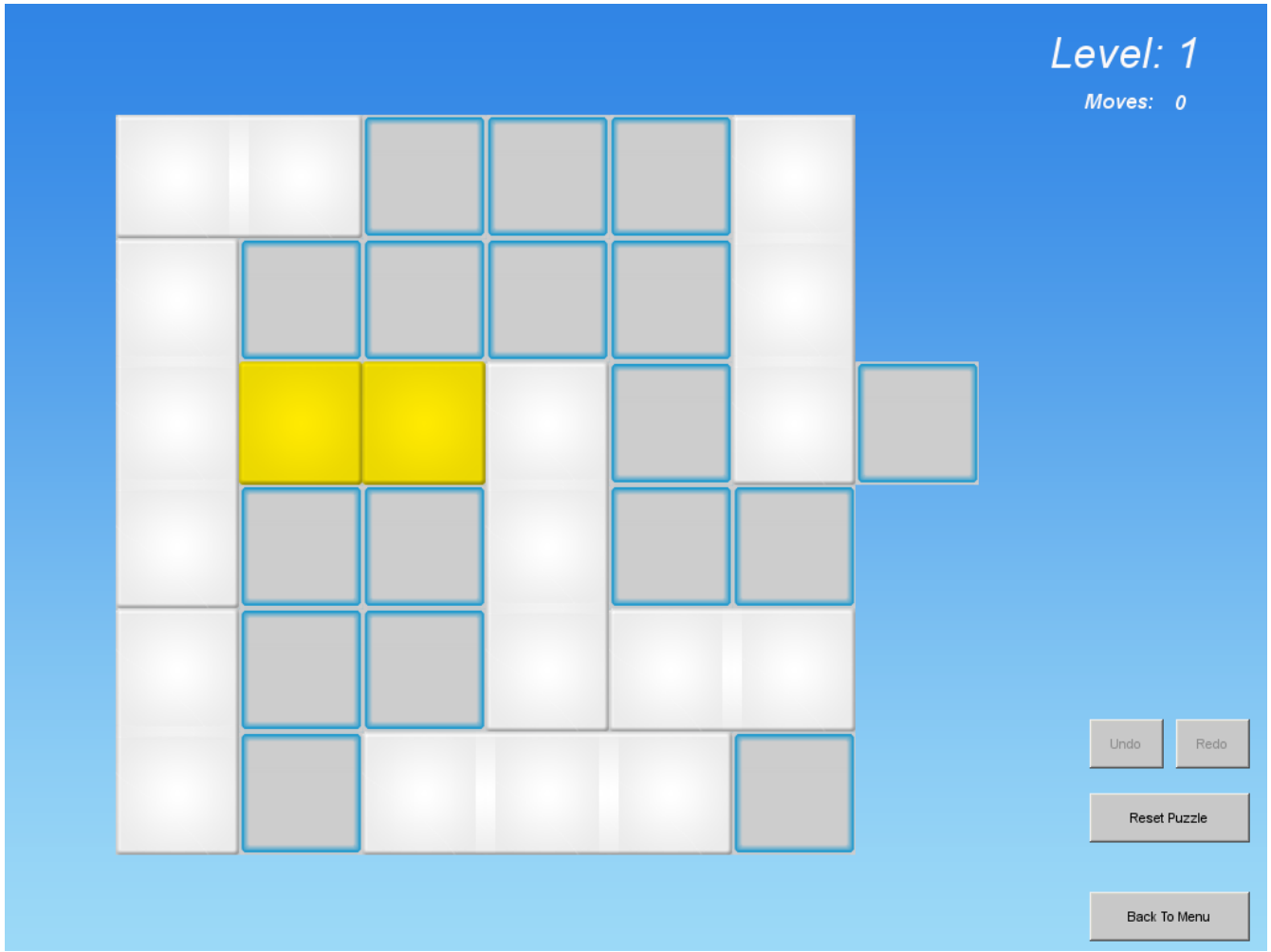


- PC-Prototyp
- DS-Technologie
  (warten auf DevKit)

PC-Prototyp

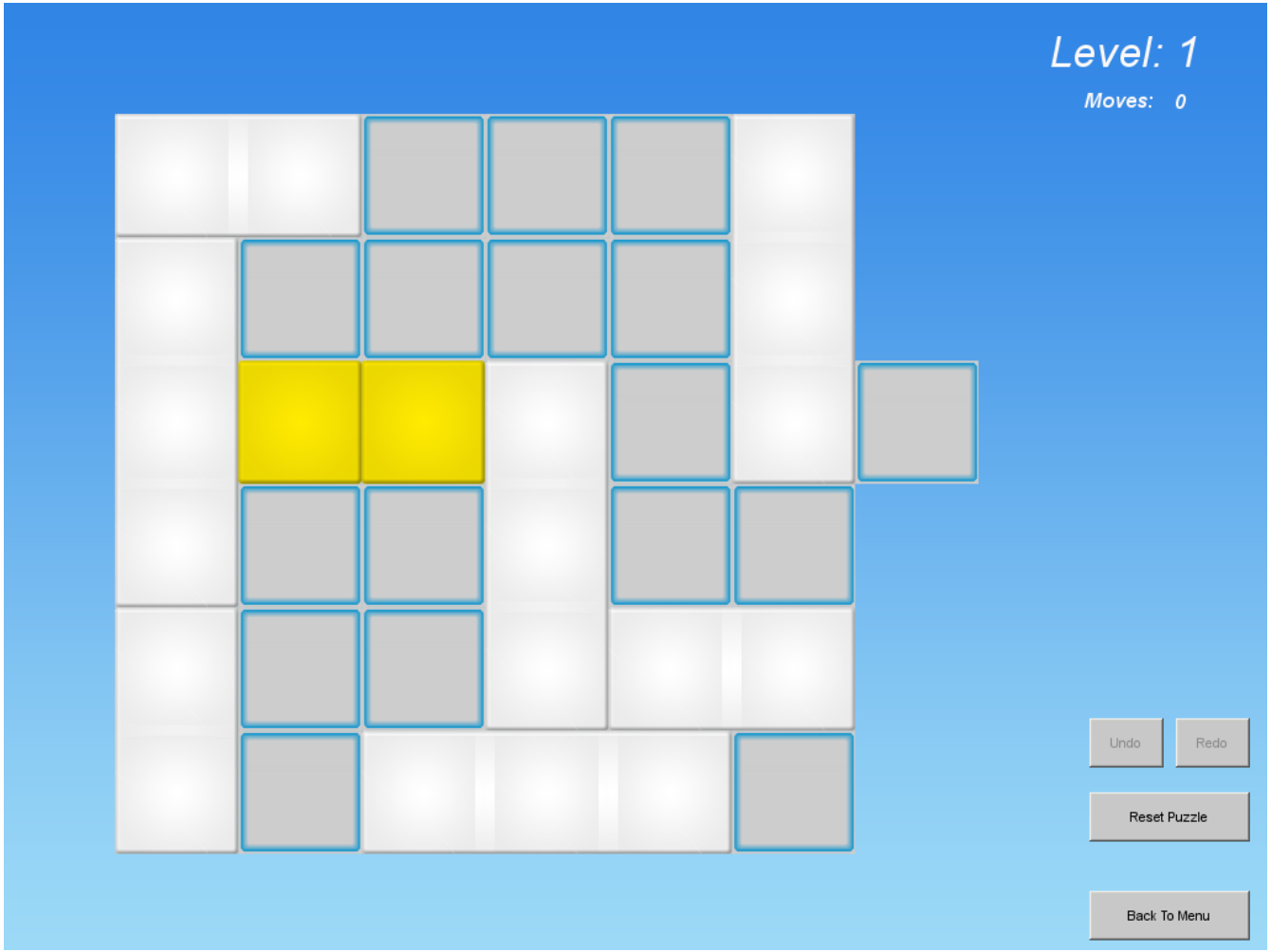
- Implementation der Spiellogik
- Basierend auf 2D-Interface-Engine
- Tests unterschiedlicher Spielvarianten
- Rudimentäre Tools (Level zunächst als Text- bzw. Xml-Files erstellt)

Erster PC-Prototyp 2D

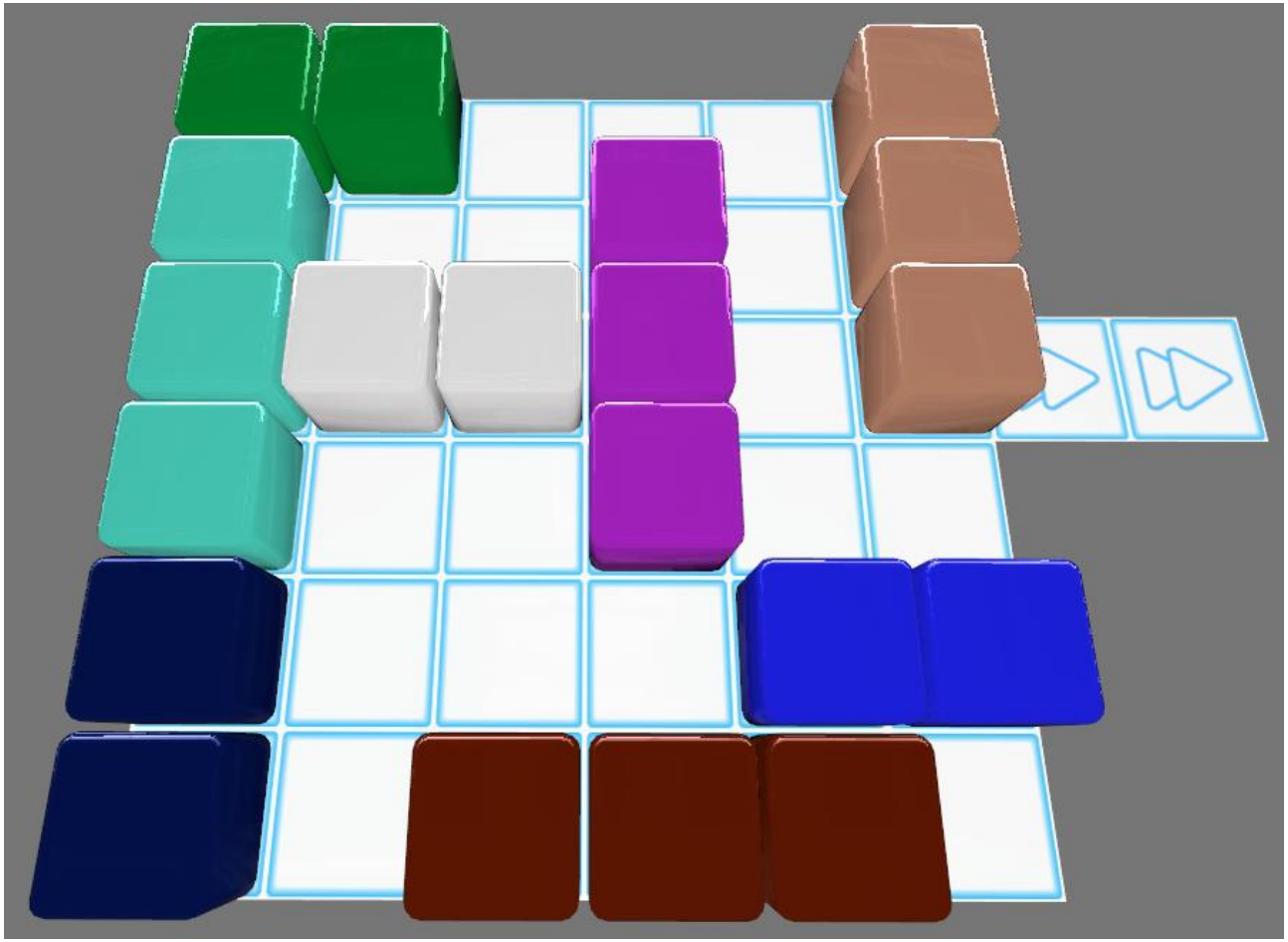


Erstes Level-Fileformat

```
11---1
2----1
2PP3-1E
2--3--
1--322
1-111-
```



Erster PC-Prototyp 3D



DS-Technologie

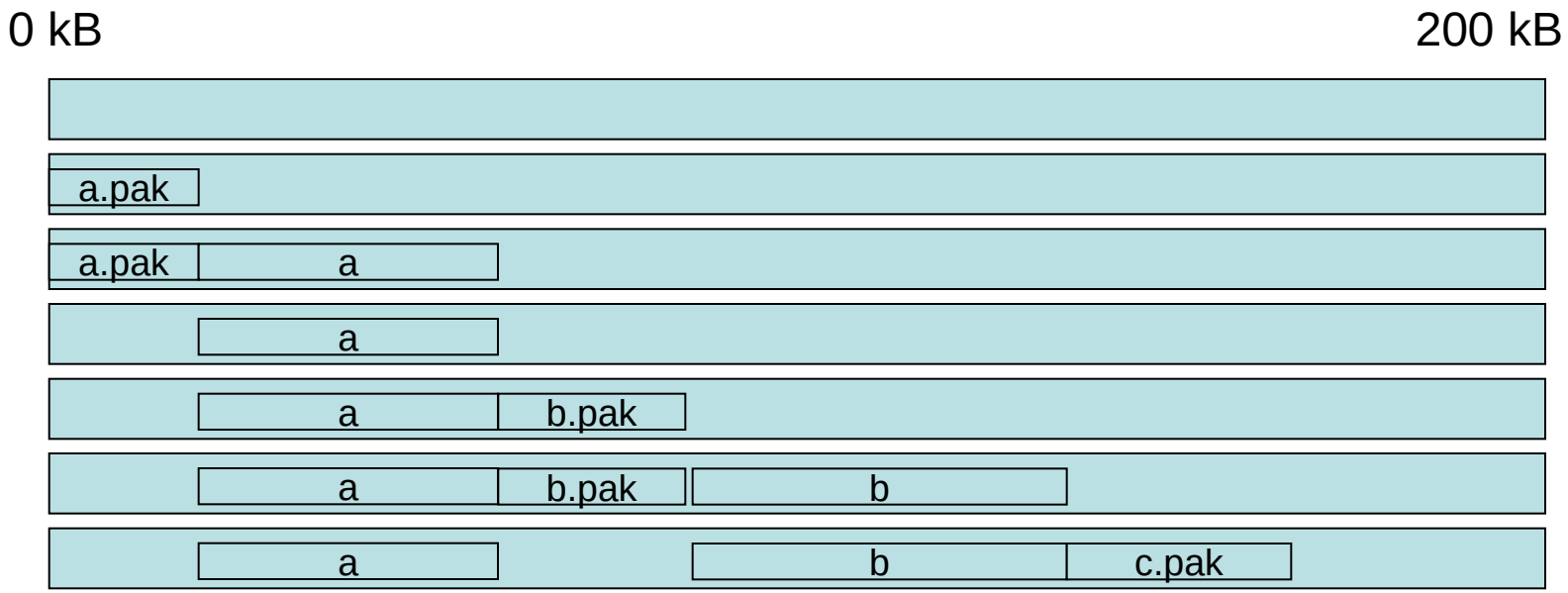
- Umsetzung des PC-Frameworks für NintendoDS
- Ziele:
  - Anbindung der Nintendo-Tool-Pipeline
  - Möglichst große Ähnlichkeit zu den (public) Interfaces der PC-Engine
- Probleme:
  - Hardware könnte unterschiedlicher nicht sein
  - geringer Haupt- und Grafikspeicher
  - Grafikhardware ist teils sehr speziell
  - keine Fließkommaberechnung in Hardware

Ressourcenverwaltung

- PC:
  - Virtueller Speicher:
    falls physikalischer Speicher voll ist, werden nicht benutzte Daten auf Festplatte geswappt
- DS:
  - Speicher ist unter Umständen sehr limitiert (insbesondere Grafikspeicher)
  - Speicherfragmentierung kann großes Problem sein

Speicherfragmentierung

- Beispiel 1:
  - Laden von 3 Ressourcen aus gepackten Dateien (40/50/60 kB, 50% Kompression, 200 kB frei)



Out of memory

(80 kB frei, aber max. 35 kB allozierbar)

Speicherfragmentierung

- Beispiel 2:
  - Alternierendes Laden von Gameplay- bzw. Grafikdaten

Nach Freigabe der Level-Daten

(66 kB frei, aber max. 22 kB allozierbar)

Lösungen

- gezielt am hinteren / vorderen Ende des Heaps allozieren
- Aufteilung auf mehrere Heaps
  - für „langlebige“ Resourcen / temporäre Daten
  - für Interfaces / Leveldaten / Grafikdaten
- Speicherallokation auf dem Papier
  - (feste Adressen für bestimmte Daten definieren)

→ DS-Engine muss dem Applikationsprogrammierer mehr Kontrolle über Speicherallokation geben

→ DS-Engine verzichtet weitestgehend auf Verwendung der STL

→ statische Create-Funktionen statt new

PC vs. DS

- CTransformNode (PC)

```
class CTransformNode : public CSceneGraphNode
{
public:

    // initialization
    static CTransformNode* Create();                                    ///< creates a transform node
    virtual void Destroy(bool bRecursive = true);                       ///< destroys a transform node

    virtual bool IsOfType(const type_info& xOther) const;               ///< tests whether this class is or is


    // children & co
    void AddChild(CSceneGraphNode* pxNode);                             ///< adds a new child node
    void RemoveChild(CSceneGraphNode* pxNode);                          ///< removes a child node

    int GetNumChildren() const;                                         ///< returns the number of children
    CSceneGraphNode* GetChildByIdx(int iChildIdx) const;                ///< returns the child by index

    ...

private:

    ...

    // attributes
    BaseLib::C3dTMatrix             m_mLocalTransform;                  ///< the local transform matrix

    std::vector<CSceneGraphNode*>   m_apChildren;                       ///< list of all children
};
```

PC vs. DS

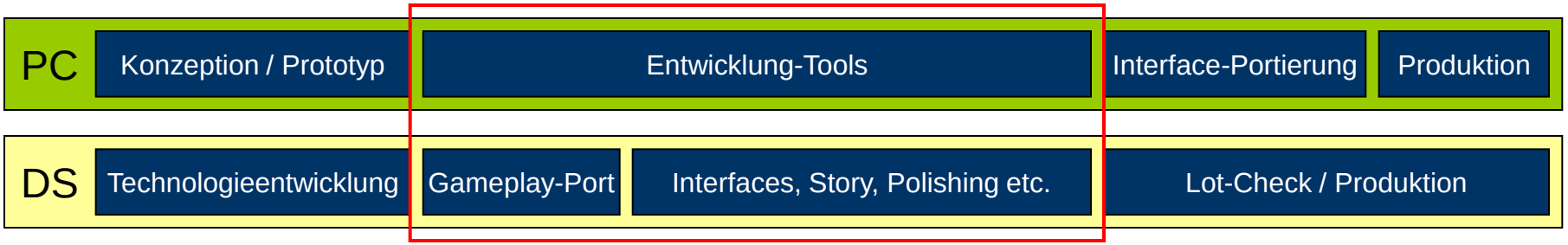
- CTransformNode (DS)

```
class CTransformNode : public CSceneGraphNode
{
public:

    DECLARE_RTTI()

    // initialization
    static CTransformNode* Create(CHeap* pxHeap = &g_xMainHeap);    ///< creates a transform node
    virtual void Destroy(bool bRecursive = true);                   ///< destroys this transform node


    // children & co
    void SetMaxChildren(u8 iMaxChildren);                           ///< sets the maximum number of children

    void AddChild(CSceneGraphNode* pxNode);                         ///< adds a new child node
    void RemoveChild(CSceneGraphNode* pxNode);                      ///< removes a child node

    u8 GetNumChildren() const;                                      ///< returns the number of children
    CSceneGraphNode* GetChildByIdx(u8 iChildIdx) const;             ///< returns the child by index

    ...


private:

    // attributes
    m44x32                          m_mLocalTransform;              ///< the local transform matrix

    CSceneGraphNode**               m_ppChildren;                   ///< array of children
    u8                              m_iNumChildren;                 ///< the number of children currently hel
    u8                              m_iMaxChildren;                 ///< maximum size of the array
};
```

Grafikhardware

- Teils sehr speziell:
  - PC-Grafikkarten rendern „nur“ Dreiecke
  - DS-Hardware hat zusätzlich 2D-Features
- Zusätzliche Klassen für DS-Features müssen implementiert werden
- Verschiedene Grafikmodi mit unterschiedlichen Fähigkeiten
  - Wechsel des Grafikmodus zur Laufzeit soll möglich sein, um Hardware optimal ausnutzen zu können

Phase 2



- DS:
  - Gameplay-Port
  - Interfaces, Grafik, Story, …
  - Weiterentwicklung Engine
- PC:
  - Programmierung Entwicklungstools

Gameplay-Portierung auf DS

- Spielprinzip benötigt keine komplexen Engine-Features
- Erstellung der ersten spielbaren DS-Version dauerte ca. 3 Tage
  - Aufsetzen der Applikation
  - Anpassungen am Gameplay-Code (ca. 2200 loc)
  - Erstellung von Dummy-Assets

Gameplay-Code-Anpassungen

- Ersetzung von PC-Datentypen (float) durch Fixkomma-Typen
  - teilweise unterschiedliche Fixkomma-Typen, je nach zu erwartender Kommaposition
- Anpassungen des MSVC-Codes an Metrowerks-C++-Compiler

- Anschließende Angleichung des PC-Codes (jedoch kein Sharing von Source-Files)

Compilerunterschiede

- enum-Typen
  - Visual Studio: inkompatibel zu int (type safe!)
  - CodeWarrior: implizite Umwandlung zu int ☹

Compilerunterschiede

- Function-Pointer
  - Visual Studio: verlangt Adress-Operator und voll qualifizierten Namen
  - Codewarrior: akzeptiert obiges, braucht aber beides nicht ☹

PC

```
CreateFunctionPointer0(this, &CGame::Start);
```

DS

```
CreateFunctionPointer0(this, Start);
```

Compilerunterschiede

- Vorzeichen von char unterschiedlich behandelt
- Unterschiedliche Regeln für implizite Typumwandlung und –truncation; zusätzliche Casts und Maskierungen notwendig

DS

```
u16
ConvertLetterUtf8ToUnicode(const char* pc)
{
          if ((u8)pc[0] >= 0xe0)
                    return (u16) (pc[0] << 12 | (pc[1] & 0x3f) << 6 | pc[2] & 0x3f);
```

PC

```
unsigned short
ConvertLetterUtf8ToUnicode(const char* pc)
{
          const unsigned char* puc = (const unsigned char*) pc;

          if (puc[0] >= 0xe0)
                    return (unsigned short) ((puc[0] << 12) & 0xFFFF |
                                             (puc[1] & 0x3f) << 6 | puc[2] & 0x3f);
```

Hardware-Unterschiede

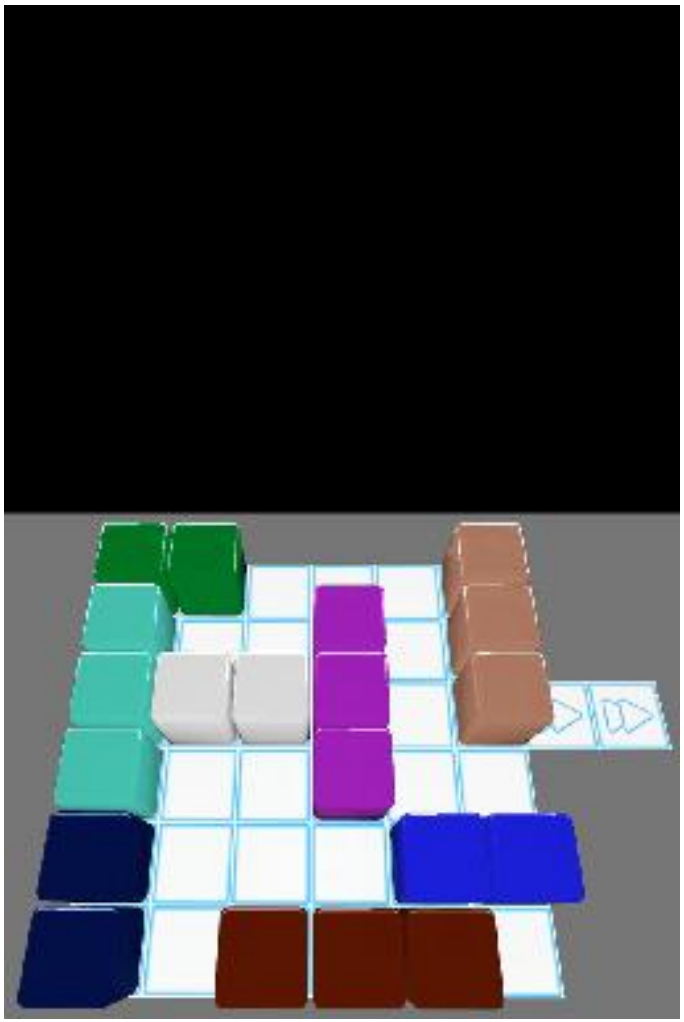
- ARM-Prozessor-Datenalignment
  - falscher Zugriff auf u32, falls Pointer nicht 4-Byte-Aligned ist

Erste DS-Version

Was fehlt noch?

- Grafiken
- Interfaces
- Sounds
- Karte / Story
- Level

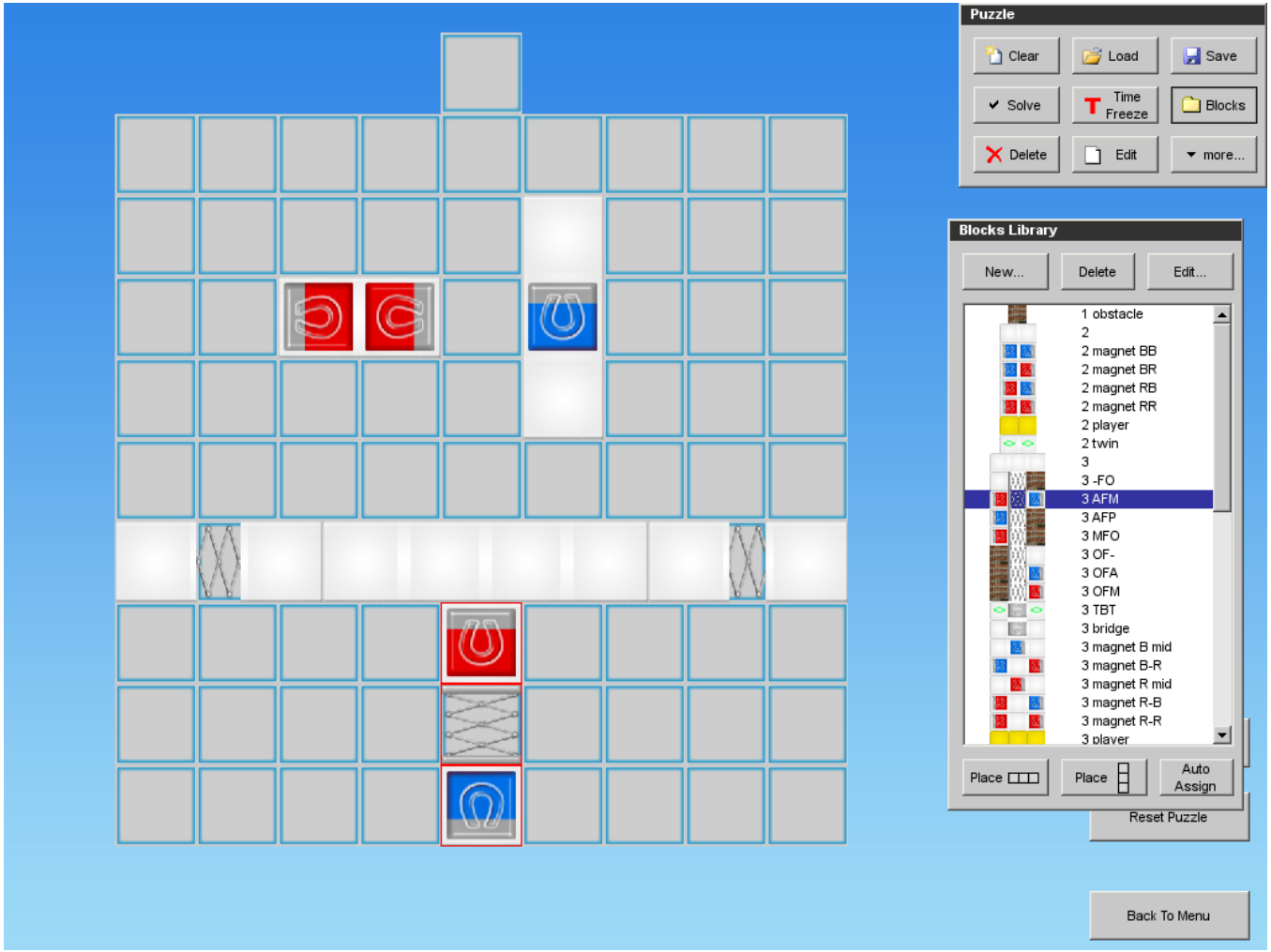
→ Unsere Arbeit für die restliche Projektdauer



PC-Entwicklungstools

- Entwicklung Leveleditor
  - Erstellung von Leveln via grafischer Oberfläche (vorher Texteditor)
  - Bibliothek von fertigen Spielblöcken
  - Automatische Zuweisung von 3D-Modellen an 2D-Level

Level-Editor auf PC



Level-Editor auf PC

PC-Entwicklungstools

- Puzzle-Solver
  - wertvolles Tool für Leveldesign
  - Berechnung von optimalen Lösungswegen zwecks Bewertung der Spielerleistung
  - Rechenkomplexität und Speichebedarf übertrifft Kapazität des DS

PC-Entwicklungstools

- Levelgenerator für einfache Level
  - Spiel verfügt über insgesamt ca. 350 Level
  - Generator erstellte Level, aus denen anschließend 200 Basic-Mode-Level gewählt wurden
  - Generierung komplexer Level einfacher als Erstellung von Hand
  - generierte Level haben anderes Spielgefühl als von Hand erstellte

PC-Entwicklungstools

- Probleme auf DS:
  - Teilweise fehlen die Ressourcen für die Ausgabe von Debug-Ausgaben
  - Crashlogs können nur mit Devkits eingesehen werden
  - Memory-Leak-Tracking ist schwieriger
- Puzzle-Testsuite
  - automatisierter Stresstest für Gameplay-Mechanik
  - Debugging auf PC deutlich einfacher als auf dem DS

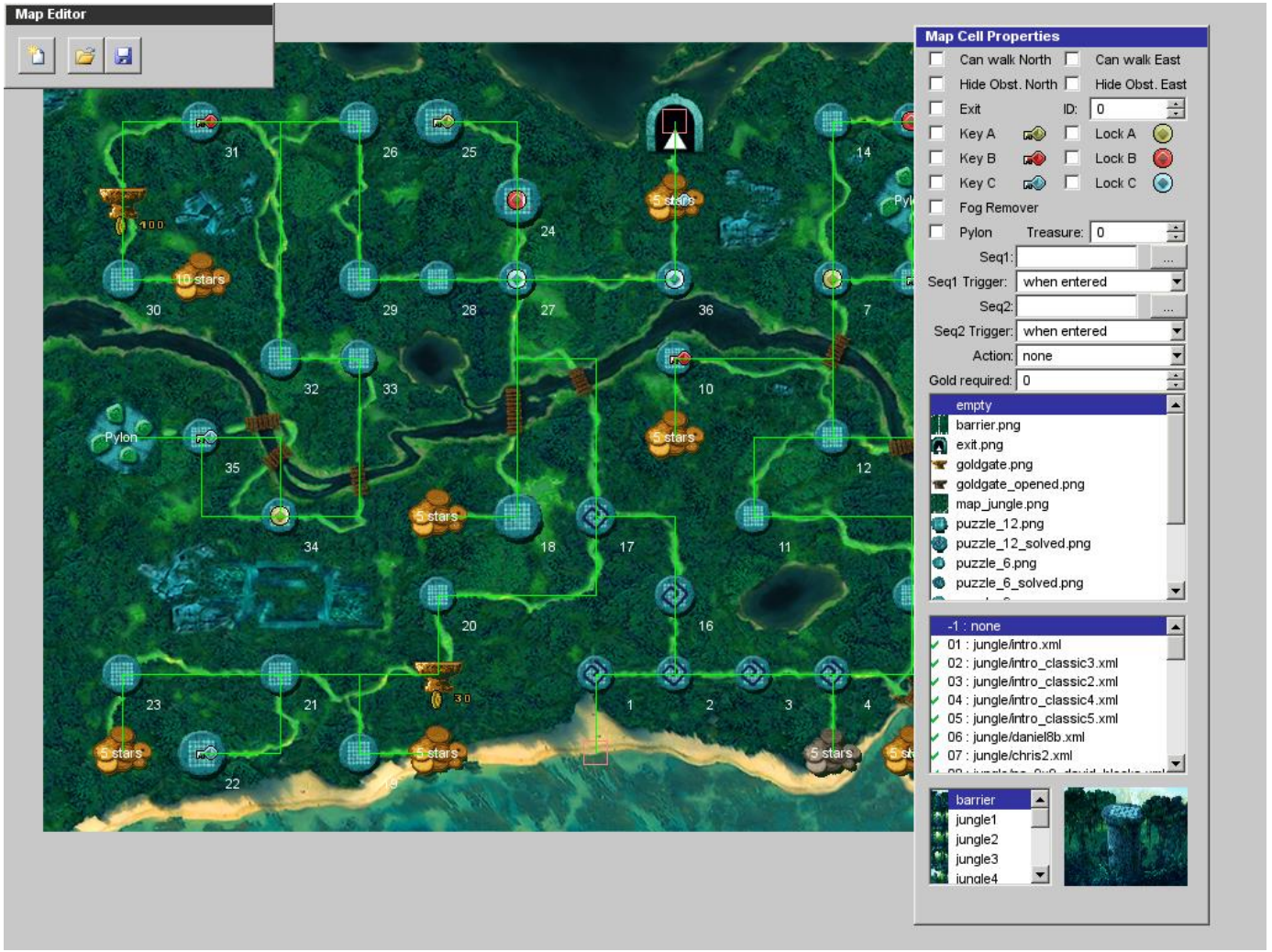
PC-Entwicklungstools

- Batch-Processor
  - bearbeitet automatisch ganze Ordner von Puzzles
  - Funktionen:
    - Berechnung der minimalen Zugzahlen
    - automatisches Zuweisen von 3D-Modellen
    - Screenshots aller Level
    - Konvertierung XML → binary

PC-Entwicklungstools

- Map-Editor
  - Erstellung der Übersichtskarte auf der sich der Spieler durch das Spiel bewegt
    - Zuweisung von Puzzles an die einzelnen Positionen
    - Definition der Wegfindungs-Informationen
    - Zuweisung von Schlüsseln / Schlössern
    - Zuweisung von Story-Cutscenes

Map-Editor auf PC



Gegenüberstellung verschiedener DS-Versionen

- Dummy-Interfaces
- Map-Implementation
- Grafiksets
- Cutscene-Player
- Sound/Musik
- Leveldesign
- Interface-Grafiken
- Saved Games, Downloadplay, Rumble-Pak, …

DS-Versionen (Mai)

- Dummy-Interfaces
  - diente zunächst als Prototyp des DS-spezifischen Interface-Systems und zur Auswahl der Spielmodi während der Entwicklung
  - Interface-Code ca. 50% des Game-Codes
  - Interface-Assets ca. 80% der Files


ROCK AND BLOCK
The secret of the living stones


... rock it!


PLAY
FILES
OPTIONS

DS-Versionen (Juni)

- Map-Implementation
  - Übersichtskarte über die einzelnen Level (Adventure-Modus)
  - bildet den Rahmen für die Story

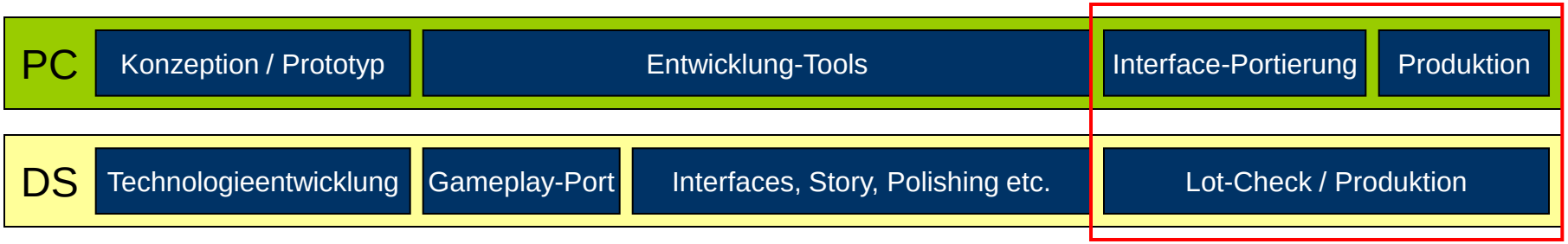
DS-Versionen (August)

- Grafiksets
  - Vier verschiedene Grafiksets für Puzzle-Darstellung
- Cutscene-Player
  - Engine-Modul zur Wiedergabe von Comic-ähnlichen Story-Sequenzen
  - Cutscene-Skripting basierend auf XML-Files

DS-Versionen (Oktober)

- Interface-Grafiken
  - Finale Interface-Grafiken in Deutsch/Englisch erstellt
- Skripting-Storycutscenes
  - 108 Cutscenes, insgesamt ca. 20 Din-A4-Seiten Text pro Sprache
- Erstellung von 150 Adventure-Mode-Leveln

Phase 3



- DS-Version:
  - Erstellung der Master-ROM-Submission-Dokumente
  - Abschließender Check durch Nintendo
  - Produktion der Module
- Rückportierung auf den PC
  - (soll im Midprice-Segment platziert werden)

PC-Version

- Stand nach Abschluss DS:
  - 2D-Prototyp-Version
  - 3D-Version zur Darstellung der NDS-Modelle
  - Gameplaycode vollständig implementiert
- Was fehlt?
  - Userinterfaces (Menüs)
  - Map / Cutscenes
  - Grafikqualität sollte für PC erhöht werden

PC-Version

- Probleme liegen eher im Asset-Bereich
  - Nintendo-DS-Interfaces sind auf 2 Screens designed worden
  - Displayauflösung DS ist 256x192
    PC-Version soll in 1024x768 laufen
  - Grafiken liegen meist nur in 512x384 vor
- Portierung des Programm-Codes vom DS zum PC weitestgehend ohne Probleme

Asset-Probleme

- Grafische Probleme wurden designtechnisch „umschifft“
  - Einrahmen von Screens der DS-Interfaces
    - Menü als Fotoalbum
    - Story-Cutscenes als Tagebuch
  - Spielkarte, die auf dem DS nur als Ausschnitt gezeigt wird, kann auf dem PC als Ganzes dargestellt werden

Interface PC vs. DS

- Spielmenu

Map PC vs. DS

- Spielkarte

Ingame PC vs. DS

Profile PC vs. DS

Programmcode

- Rückportierung DS → PC relativ einfach...
- teilweise unnötige API-Unterschiede (unsere eigene Schuld!)
- Code wird auf PC oft kürzer und einfacher
  - ... da Hardware weniger restriktiv ist
- Andererseits: heterogene PC-Hardware
  - Einstellmöglichkeiten für Graphikeffekte nötig

Programmcode

- Schwierig: Zweck der PC-Version ändert sich!
  - bisher: Toolkit + Testumgebung
  - jetzt: volle Spielversion
  - erfordert große Umbauten...
  - ... d.h. Tools gehen zeitweise kaputt
  - aufwändig, alte Features (Editoren, Tests) und neue Features (Menus, Story) parallel zu betreiben

What went wrong?

- teilweise Features integriert, die sich wirtschaftlich nicht gerechnet haben
- Map-System wurde komplexer als ursprünglich geplant
- Projekt dauerte länger als ursprünglich geplant (war aber dennoch ein finanzieller Erfolg)
- Puzzlesystem viel komplizierter als gedacht; viele langwierige Gameplay-Bugs

What went wrong?

- Übersetzungen auf DS nicht clever angegangen
  - Spiel wird in fünf Sprachen ausgeliefert
  - viele graphische Schriften, d.h. komplette Bitmaps müssen ausgetauscht werden
  - Arbeitsaufwand + Speicherplatz auf dem Modul
  - ... auf PC cleverer: Fonts und String-Tabellen benutzt (naja: GUI auf PC viel leistungsfähiger!)

What went right?

- Deutliche Synergien zwischen PC- und DS-Version
- Mit dem Projekt wurde Technologie geschaffen, die auch in kommenden Projekten eingesetzt werden kann
- Zusätzliche Einnahmen durch Technologielizenzierung an andere Entwicklungsstudios
- Eigenfinanzierung ermöglichte Entwicklung ohne externen Druck sowie Besitz der IP

Fazit

- Prototyping auf PC, Entwicklung auf DS, Rückportierung auf PC ist für Programmierung durchaus gangbarer Weg

- falls PC-Version von Anfang an geplant ist, wäre Asseterstellung für PC und Reduktion zum DS möglicherweise besser

Fragen?