Satori

arkanoid

Programátorský manuál

Obsah

1. Program

Dokumentace popisuje logicko-akční hru typu arkanoid. Ve hře postupuje hráč jednotlivými levely kde pro dohrání levelu musí zničit všechny cihly. K tomu má jeden až maximálně třicet balónů, které odráží jednou nebo několika pálkami ovládaných myší.

Na rozdíl od ostatních klonů arkanoidu nejsou pozice kostek diskrétní, stěny levelu ani pálka nejsou na stále stejném místě. Každý level je vlastně takový vektorový obrázek ve kterém mají polygony funkce pálek, zdí, cihel apod.

1.1 Cíle hry

Primární :

1. Dokončit poslední level hry a tím hru dohrát. Hra obsahuje více cest přes různé úrovně a tak není nutné dokončit všechny levely.
2. Dokončit aktuální úroveň a přesunout se tak do další. Po ukončení některých úrovní se hráči nabídne výběr dalšího levelu. Během hry je také možné získat vstup do jinak nepřístupných bonusových levelů.

Sekundární :

1. Nahrát vysoké skóre a zapsat se do skóre tabulky deseti nejlepších.
2. Dohrát celou hru v co nejkratším čase. Předpokládá perfektní zvládnutí všech úrovní.
3. Dohrát úroveň v co nejkratším čase.
4. Během každé úrovně nahrát co největší skóre.
5. Dohrát hru všemi možnými cestami větvení úrovní.

1.2 Požadavky

Operační systém Windows 95/98/NT/2000/XP s OpenGL 1.1. Rychlost nejvíce ovlivní velikost paměti grafického akcelerátoru (alespoň 4MB, doporučeno 64).

Hratelné :

20 fps (640x480)

procesor Intel II 233.0 Mhz, 96 MB RAM (SDR)

grafický akcelerátor ATI , 4MB RAM, AGP x2

150 fps (640x480)

procesor AMD 1.0 Ghz, 256 MB RAM (SDR)

grafický akcelerátor Riva TNT2, 32MB RAM, AGP x4

600 fps (640x480)

procesor Intel Pentium 4 2.4 Ghz, 512 MB RAM (RIMM)

grafický akcelerátor G4 Ti4200, 64 MB RAM, AGP x4

Hru je do budoucna možné zkompilovat na systémy:

SGI, IRIX 5.2 a vyšší

DEC Alpha, OSF/1 a OpenVMS

IBM RS/6000, AIX

Linux

Unix

1.3 Instalace

Hra si neukládá jakékoliv informace jinam než do adresáře, ze kterého byla spuštěna. Díky tomu ji stačí kopírovat.

1.4 Ovládání

Hra se ovládá klávesnicí a převážně myší. Kromě zadávání jména pro zápis do skóre tabulky stačí myš.

1.5 Grafika

Celá hra běží na 3D enginu pod OpenGL. Každý snímek je vykreslen v jedné rovině 3D prostoru a 32bitové textury si žádají spoustu paměti grafické karty. Výhodami jsou rozlišení od 640x480 po 1600x1200, 32bitová hloubka barev, alfa míchání, barvy materiálů a snadnější naplnění Moorova zákonu.

1.6 Zvuky

Zvukové prostředí zajišťuje knihovna **fmod.dll**. Podporovány by měly být všechny zvukové karty. 16bitové vzorky jsou samplované od 4000 do 48000 Hz. Přehrávání probíhá v 3D prostoru a zvuky tak mají hlasitost podle vzálenosti od pálky „posluchače“ včetně Doplerova jevu z rychlosti pálky.

1.7 Soubory

Hra se spouští přes soubor **satori.exe** pro plnou funkčnost jsou vyžadovány **data.w1**, **sound.w1** a dynamické knihovny (dll) : **fmod.dll**, **freeType-6.dll** a **ijl15.dll**. Soubory **score** a **setup.ini** si umí hra znovu vygenerovat, ale i tak je lepší je nemazat.

Pro uložení nové konfigurace hry nebo zápis do skóre tabulky je nutné mít práva zápisu a spouštět hru z média na které lze zapisovat.

2. Části programu

2.1 Knihovny

2.1.1 textures.ini

Knihovna textur grafického uživatelského rozhraní GUI. Nahraná po spuštění programu. V paměti do jeho ukončení. Kurzor, tlačítka, pozadí menu, obrázek při nahrávání hry, okna. Knihovna bude znovu nahrána při aplikaci nového rozlišení.

Záznam :

Id : INT identifikace textury

SubId : INT větší než nula označuje id hlavní textury (s více subtexturami)

SubX : INT pozice subtextury X v souřadnicích hlavní textury, jinak 0

SubY : INT pozice subtextury Y v souřadnicích hlavní textury, jinak 0

W : INT šířka textury nebo subtextury

H : INT výška textury nebo subtextury

Path : STRING místo uložení textury

AlphaMode : INT způsob výpočtu alfa kanálu textur, u textur s uloženou alfou jako jsou 32 bitové targy (*.tga) nemá vliv

0 … neprůhlená textura = každý texel má hodnotu 255

1 … při součtu r+g+b menším 10 průhledná jinak neprůhledná, výchozí

2 … výpočet alfy podle intenzity texelů

Frames : INT počet snímků na generování

IncX : INT přírustek X pro generování animovaných subtextur

IncY : INT přírustek Y pro generování animovaných subtextur

FrameTime : INT doba zobrazení jednoho snímku

Loop : INT příznak cyklení = 1, vygenerované animace cyklí pořád bez ohledu

ExternFrames : INT počet textur nasledujících za toutu, tvořících dohromady jeden animační celek, pro použití na objekty řízenou animaci

Příklad:

Z jednoho obrázku effect01.tga vygeneruje 4 animace po čtyřech snímcích ale do paměti je nahraje jako jednu texturu 256 x 256.

```
define tex001 {  Id "1"   SubId "0"   SubX "0"   SubY "0"    W "256" H "256"
                  Path "textures/set1/effect01.tga"
                  Frames "4" IncX "64"  IncY "64" FrameTime "100" Loop "1"  }

define tex002 {  Id "2"   SubId "1"   SubX "0"   SubY "64"   W "256" H "256"
                  Path "textures/set1/effect01.tga"
                 Frames "4" IncX "64"  IncY "64" FrameTime "100" Loop "1"  }

define tex003 {  Id "3"   SubId "1"   SubX "0"   SubY "128"  W "256" H "256"
                  Path "textures/set1/effect01.tga"
                  Frames "4" IncX "64"  IncY "64" FrameTime "100" Loop "1"  }

define tex004 {  Id "4"   SubId "1"   SubX "0"   SubY "192"  W "256" H "256"
                  Path "textures/set1/effect01.tga"
                  Frames "4" IncX "64"  IncY "64" FrameTime "100" Loop "1"  }
```

2.1.2 sound.ini

Knihovna zvuků a hudby GUI a základních (společných) zvuků pro levely. Nahraná po spuštění programu. V paměti do jeho ukončení. Zvuky lze přehrávat z libovolné části hry uvedením jejich indexu. Dva vnořené adresáře **samples** a **music**.

Záznam :

id : INT identifikace zvuku nebo hudby

Path : STRING místo uložení

loop : BOOL příznak opakování zvuku = 1, hudba se opakuje automaticky

Příklad:

```
samples {
   sample001 { id "500"   path "soundrooms/room01/Lick1.ogg" }
   sample002 { id "501"   path "soundrooms/room01/Lick2.ogg" }
   sample003 { id "502"   path "soundrooms/room01/Lick3.ogg" }
}

music {
   music002  { id "506"   path "music/alien.mid" }
}
```

2.1.3 levelsetXY.ini

Knihovna textur společná pro více levelů spojených do jednoho světa. Nahraná při prvním levelu, který ji vyžaduje (proto se první level nahrává poprvé tak dlouho). V paměti dokud některý z levelů nepožádá o jiný levelset. Animace efektů, pálek, výbuchů, balónů, částic a jiných prvků zdílených mezi více levely. Stejný formát jako textures.ini.

2.1.4 levels.ini

Informace o všech levelech hry. Nahraná po spuštění hry. V paměti až do ukončení. Počet záznamů je počtem levelů hry. Je dobré číslovat levely level001, level002... bez vynechávání (jsou načítány jako level%d kde %d je číslo levelu - prozatím)

Záznam :

id : INT identifikace levelu

name : STRING název levelu zobrazený při nahrávání

textures : STRING jméno knihovny levelset

textures_ext : STRING jméno rozšiřující knihovny textur

soundroom : STRING jméno knihovny zvuků

data : STRING jméno souboru s definicí levelu

lastLevel : INT příznak posledního levelu hry, nasleduje OUTRO, hra muze mit libovolný počet zakončení

Příklad:

```
define level001 {
   id          "1"
   name      "practice"
   textures      "system/levelset1.ini"
   textures_ext  "system/level01.tex"
   soundroom "system/soundroom01.ini"
   data      "levels/level01.ini"
}
```

2.1.5 soundroomXY.ini

Knihovna lokálních zvuků a hudby jedné úrovně. Obvykle nesdílená více levely. Nahraná při nahrávání levelu, který ji vyžaduje. V paměti dokud jiný level nepožaduje odlišnou knihovnu soundroom. Zvuky lze přehrávat pouze v levelu uvedením jejich indexu. Stejný formát jako **sound.ini**, ale podstatně méně záznamů !

2.1.6 levelXY.tex

Knihovna lokálních textur jedné úrovně. Obvykle nesdílená více levely. Nahraná při nahrávání levelu, který ji vyžaduje. V paměti dokud jiný level nepožaduje odlišnou knihovnu (*.tex). Textury lze využívat jen v levelu. Stejný formát jako textures.ini a levelset.ini, ale podstatně méně záznamů ! V základu pouze obrázek pro nahrávání levelu, velké pozadí levelu (přes celý obraz) a některé unikátní textury. Pro přístup k texturám knihovny objekty levelu (pozadí, zeď, pálka...) zvýšit id o 5000.

Příklad: s jednou texturou v knihovně a definicí levelu

soubor knihovny level01.tex...

```
define tex01 {  Id "1"   SubId "0"   SubX "0"   SubY "0"    W "256" H "256"   Path
"textures/level/level001/level01.jpg" }
```

soubor level01.ini...

```
define level {
    id  "1"
    loadingImg  "1"    // pristup ke knihovne *.tex s indexem 1
    ..
          // stejna textura z knihovny *.tex, index o 5000 vetsi
    bgr001 { tex "5001" pos "0.0, 0.0, 0.0" width "800.0" heigth "600" }
}
```

2.2 Zdroje dat

2.2.1 Soubor w1

Formát souboru (*.w1) vznikl z potřeby zakódování dat hry s vyloučením jejich modifikace zkušenějšími uživateli (usnadňování levelů a tak celé hry). Další cíl byl dosáhnout lepší komprese než u formátu (*.zip), kterého bylo dosaženo zlepšením komprese o 10 % oproti zip.

Soubor (*.w1) obsahuje strukturu podadresářů s jejich soubory a kontrolními součty. Ty umožňují detekci neoprávněných změn.

Nevýhodou je extrémně pomalá komprese souborů do formátu w1. Dekomprese je také pomalejší (ale podstatně rychlejší oproti kompresi) než u zip (soubory je nejen nutné dekomprimovat ale i dekódovat hned v několika vrstvách).

2.2.2 Textury

Vlastní Zone engine, na kterém celá hra běží podporuje následující formáty. Jejich rozměry musí být vždy 2^x tedy 8,16, 32, 64, 128, 256, 512, 1024. Daly by se podporovat i jiné rozměry a po nahrání je převézt na nejbližší 2^x rozměr.

Díky implementaci subtextur lze do jedné textury 2^x naskládat více obrázků, které už mohou mít libovolné rozměry. Subtextury by měly snižovat paměťové operace kde místo dvaci menších textur stačí jedna větší obsahující těchto dvacet subtextur. Problémem by mohly být osmi bitové textury s paletami, u kterých skládání do většího celku ničí kvalitu barev omezenou paletou.

JPeG - (*.jpg), podpora přes knihovnu **ijl15.dll,** větší textury se ztrátou kvality typem komprese neumožňují přesný výpočet alfa kanálu

TarGA - (*.tga), 8, 16, 24 a 32 bitů, komprimovaný i nekomprimovaný, **nejlépe** navržený **bezeztrátový** grafický formát, nejširší podpora

GIF - (*.gif), 8 bitů, komprese LZW, verze GIF87a

PCX - (*.pcx), 8 bitů, komprese RLE, 24 bitová verze není zatím podporována, nejlépe převézt na TGA.

2.2.3 Zvuky a hudba

Je možné použít všechny formáty podporované knihovnou **fmod.dll**. Samply budou při nahrávání převedeny na **MONO** kvůli jejich provozu v 3D prostředí. Nejvhodnější typ je .OGG 16-bit mono. Z pohledu přístupu ke zvukům a hudbě z levelů, který se provádí uvedením identifikačního čísla, splývají pojmy hudba a samply v jeden. Všude kde může znít hudba lze pustit i samply a obráceně. Výbuchy, střelba nebo jiné zvuky typické pro samply mohou být i krátké treky v MIDI.

Hudba - **.MOD**, **.S3M**, **.XM**, **.IT** nebo **.MID**

Samply - **.WAV**, **.MP2**, **.MP3**, **.OGG** nebo **.RAW**

2.2.4 Fonty

Zone engine poskytuje tři typy fontů.

GLUT - fonty aplikačního rámce, výhoda – jsou použitelné kdykoliv, použití při nahrávání hry nebo levelu kdy ještě nejsou načtené ostatní fonty

TgaFont - sada písmen v jedné textuře, stejné rozměry písmen, pracné vytváření

TrueType - podpora knihovnou FreeType-6.dll, výborná kvalita, lze použít většinu (*.ttf) fontů, možné komplikace v některých státech kde si firma Apple dala patentovat mechanizmy bajtových kódů

2.3 Formát defineTab

Si lze představit jako adresářový strom kde každý soubor představuje proměnou a jeho obsah její hodnotu. Zápis je podobný syntaxi C++. Kromě textur, spustitelného exe souboru, dynamických knihoven, fontů a hudebních vzorků je v něm uložena celá hra. Ze souboru nadefinovaná teabulka se umí po změně hodnot (setup) znovu uložit včetně poznámek.

Problematická je absence jakýchkoliv kontrol správného zápisu. Pro porušení celé tabulky stačí vynechat jedinou "závorku a zbytek kódu se stane opravdu dlouhou hodnotou této proměné.

2.3.1 Klíčová slova

Tabulku tvoří proměné a příkazy. Každé proměné se přiřadí hodnota uzavřená do "závorek" (pro podporu hodnot s mezerami). Po definici proměných proběhne postupné vykonávání příkazů podle požadavků objektu, který si její definici vyžádal. Klíčové slovo **define** není třeba uvádět, ale v některých případech kdy se proměná jmenuje stejně jako některé klíčové slovo je to nutné a může i vylepšit čitelnost kódu.

Preprocesor :

#include "soubor"

#define { tělo které bude vloženo }

Ostatní :

define proměná "hodnota"

load typ_objektu proměná

alloc typ_objektu počet

do, add, component, msg : pro skriptování GUI, v této hře mrtvá klíčová slova

2.3.1.1 #include

Include stejně jako u C/C++ vloží namísto #include "jmeno_souboru" textový soubor tohoto jména. Není tak nutné neustále opakovat definici pálky, která je stejná pro několik úrovní. Vložení je omezeno na jednu úroveň zanoření z důvodu přehlednosti.

2.3.1.2 #define

#define funguje jako jednoché makro. Jeho tělo se uzavírá do { závorek },které jsou při v kládání na místo uvedení jména makra kdekoliv ve stejném souboru odstraněny. I jednočíselné jednořádkové makro musí být takto ozávorkováno.

2.3.1.3 define

Define přiřazuje proměné hodnotu nebo více hodnot najednou tj vytvoří adresář proměných. Pro vytvoření adresáře stačí proměné { ozávorkovat }. Adresáře lze libovlně vnořovat.

```
Příklad : stejný účinek
define color “white“
color “white“

Přiklad 2 : adresáře
define colors {
   color1  “white“
   color2  “red“
   color3  “blue“
}
```

2.3.2 Poznámky

Je možné využít poznámky jako C/C++. Tedy dvě lomítka // pro poznámku do konce řádku. Nebo několika řádkovou poznáku uzavřenou do /* poznáka */. Několika řádková poznáka končí nalezením prvního dvojznaku */ pokud nepatří k další zanořené poznámkové dvojici.

Poznámky jsou odstraněny v prvním kroku preprocesoru a jsou nahrazeny jednou mezerou.

```
Příklad:
define color “yelow“  // poznamka.....
define /*  poznamka
  na vice řádků  /* s další vnořenou */
*
*/
color2 “purple“
```

3. Efekty

40 základních efektů + 2 kombo efekty

Efekty (bonusy) padají z rozbitých cihel a bonusových generátorů. Mohou mít i okamžitý účinek rozbitím cihly bez nutnosti vypadnutí a braní (kolizí s pálkou). Do cihel se umísťují většinou s náhodnou pravděpodobností výskytu.

Efekt ovlivňující pálku, získá ta pálka která jej vezme. Ostatní efekty lze brát, kteroukoliv pálkou (pokud je jich víc).

Sebráním efektu ovlivňující balón získá jeho účinek vždy jen jeden z balónů (pokud je jich víc). Přenesení tohoto účinku na více balónů je možné jen rozdvojením, roztrojením nebo rozpětcením (množením balónu). Množení balónů je také jediný efekt, který proběhne na všech balónech v úrovni.

Příklad : máte jeden balón a získáte efekt rozdvojení = máte dva balóny

máte dva balóny a získáte efekt roztrojení = máte šest balónů

3.1 Základní efekty

	Globální efekty
	náhodný efekt
	náhodně poškodí všechny kostky v levelu
	sníží osvětlení levelu
	zvýší osvětlení levelu
	náhodný efekt ovlivňující pálky a náhodný efekt pro míče
	síť
	přehodí všechny spínače v levelu
	zrychlení času (kumulující, při druhém sebrání zrychí ještě více)
	zpomalení času

	Efekty balónů
	**Velký** balón
	Balón **vystřelí** čtyři rakety
	**Dvojnásobné skóre** za rozbité kostky
	Létá **skrz kostky**, zvyšuje **skóre**
	Rozbíjí vše na jednu ránu, zvyšuje **skóre**
	**zrychlení**
	**zpomalení**
	**Střílející míč** Zesílení – více směrů střílení
	Při prvním nárazu do kostky jednou **vybuchne**
	**Neviditelný** míč Zesílení – nuluje efekt
	**Rozdvojení** všech balónů
	**Roztrojení** všech balónů
	**Rozpětcení** všech balónů

	Efekty pálek
	**Laser**, 5 střel
	**Kulomet**, 5000 střel Zesílení – počet vystřelených projektilů najednou
	**Raketa**, 2 střely (exploze) Zesílení – počet vystřelených raket najednou
	**Mina**, 5 střel (exploze po nárazu balónu) Zesílení – počet vystřelených min najednou
	**Ztráta života**
	**Život navíc**
	**Lepidlo** (magnet) Zesilování maxima přilepených balónů (1, 4, 9, 16 …)
	**Odpuzující pole** Zesilování síly
	**Přitažlivé pole** Zesilování síly
	**Obrácené ovládání pálky** Dvakrát po sobě nuluje efekt
	**Zvětšení** Zvětšení až dvakrát
	**Zmenšení** Zmenšení až dvakrát

	Bonusové efekty
	Přidá **50** skóre bodů
	Přidá **5%** aktuálního skóre

	Efekty s levely
	**Restart levelu**
	**Další level**
	**Vstup do bonusového levelu**

	Globální efekty
	Zvýší **sílu** používaných efektů, pokud žádné nejsou uschová se
	**Bonus**, za každých 50 nasbíraných je jeden život

3.2 Kombo efekty

Sebráním některých dvou efektů po sobě přidá speciální funkci.

Odemikací balón : Po sebrání **FEST** a **E** nebo obráceně. Poslední efekt zůstává.

Balón střílí rakety : Během střílejícího balónu (**GUN** ball) vzít Bomby (**BOMB** ball).

3.3 Užívání síly

Pokud je uschovaný nepoužitý efekt síly (**P**) lze sebráním dalšího efektu vyvolat jeho odlišné chování.

BFG : proběhne dvakrát po sobě

Přidání života : přidá dva životy

Rozsvícení : bude účinnější

Bonus +5 % : bonus + 10%

Zvyšování efektu balónů obecně zvyšuje skóre za jimi rozbité kostky.

4. Level

4.1 Co je level

Level tvoří množina objektů prezentovaných dvou-rozměrnými bitmapami s definovanými kolizními tvary, které se skládají z polynomů a kruhů. Objektu se přiřazuje funkce : pozadí, zeď, pálka, balón, pole , generátory efektů, přepínače, kostky.

Postupu na další úroveň nebo vytvoření nové hry vyvolá náhrání úrovně. Nahrávají se jen ty knihovny, kterými se liší od předchozí úrovně. Při vytvoření hry se musí nahrát všechny knihovny využívané prvním levelem. Během nahrávání je zobrazen obrázek **loadingImg** a krátký text **LevelLoadingText**. Po nahrání se začne přehrávat hudba **levelMusicId** (pokud je definovaná) a uživatel je vyzván k zahájení hry stiskem klávesy nebo tlačítka na myši.

Po zahájení hry ovládá uživatel pálky a balónem ničí cihly. Ve hře je možný návrat do menu stisknutím tlačítka ESC pro změnu nastavení, pauzování hry nebo její ukončení. Zničením všech cihel nebo sebráním efektu **LEVEL** je úroveň úspěšně dokončena. V průběhu je možné sebrat efekt **RESTART**, který vyvolá opětovné nahrání levelu nebo efekt **BONUS** pro vstup do bonusové úrovně (otevře fork po dončení úrovně). Při dokončení způsom ničení všech cihel záleží na nastavení **GAME – next level** kde v případě volby **user** hra čeká na stisk klávesy nebo tlačítka a tím hráči umožní posbírat všechny padající bonusy (čas je v tomto bodě zastaven) nebo volbou **auto** automaticky přejde k zobrazení výsledků.

Dokončení zobrazí dosažené rekordy nejkratšího času a nejvyššího skóré nahraného na této úrovni. Dále skóre získané za dokončení úrovně (to závisí na serii po sobě dohraných levelů beze ztráty života) a skóre za čas do vypršení časového limitu pokud byl na level stanoven. Uživatel je vyzván ke sticku klávesy nebo v případě větvení cesty levely z této úrovně jsou zobrazeny dva obrázky **forkImg1** a **forkImg2** s krátkými popisky **forkText1** a **forkText2,** ze kterých uživatel jeden vybere klepnutím myší. K číslu aktuální úrovně je podle toho přičteno **fork1AddId** nebo **fork2AddId**.

Level je uložen a editován ve formátu **defineTab** v textových souborech s nejčastější příponou (*.INI). Tento způsob prakticky znemožňuje vytvoření editoru a proto s heslem „editory vytváří stereotypní hru“ nezbývá než každý level editovat ručně.

4.2 Vlastnosti levelu

Při ne definici symbolů budou jejich proměnné nastaveny na výchozí hodnoty 0.

LevelLoadingText :

STRING

text zobrazený během nahrávání levelu, řádky oddělené znakem „ / “ maximálně 10 řádků po 255 znacích

id :

INT

identifikace levelu (číslo úrovně pro kontrolu).

loadingImg :

INT

Obrázek zobrazený během nahrávání levelu.

nextLevelAddId :

INT

Zvýší id levelu pro skok na další úroveň

výchozí 1

nextBonusLevelAddId :

INT

Zvýší id levelu pro skok na bonusovou úroveň

výchozí 1

forkImg1 :

INT

Index obrázku (knihovna tex) pro první cestu.

forkImg2 :

INT

Index obrázku (knihovna tex) pro druhou cestu.

forkText1 :

STRING

Text pro první cestu.

forkText2 :

STRING

Text pro druhou cestu.

fork1AddId :

INT

Zvýší id levelu pro skok na další úroveň první cesty

fork2AddId :

INT

Zvýší id levelu pro skok na další úroveň druhécesty

levelTimeLimit :

INT

časové omezení na dohrání levelu (v sekundách)

levelMusicId :

INT

Hudba přehrávaná během levelu (*.wav, *.mid, *.s3m, *.mp3, *.ogg) ze zvukové knihovny soundroom.

Výchozí 0 = žádná hudba

nextBonusLevel :

INT

Z levelu je možné vstoupit do bonusového kola. 1=ano

bonusLevel :

INT

Označuje bonusový level. 1=bonusový

levelBgrEnv :

INT

Typ grafického pozadí. 0=žádné, 1=padací hvězdy, 2=déšť

4.3 Proměnné prostředí

gravitation :

VECTOR

Gravitace levelu. Ovlivňující balóny a padající bonusy.

env_velocity :

VECTOR

Počáteční vector pro padající bonus.

env_speed :

DOUBLE

Faktor ovlivňující rychlosti.

env_mass :

DOUBLE

Faktor ovlivňující hmotnosti.

levelBgrEnv :

INT

Typ grafického pozadí. 0=žádné, 1=padací hvězdy, 2=déšť

4.4 Definice soundroom

Soundroom je soubor zvuků přehrávaní během levelu. Vnořený adresář

define SoundRoom { ….} obsahuje definice zvuků :

sound { int iID, int iTimeMin, int iTimeMax, int iPercent=100, int iVolumePercent=100 }

iID : Index zvuku z knihovny soundroom

iTimeMin : Minimální počet tisícin (1s = 1000) pro spuštění zvuku

iTimeMax : Maximální počet tisícin (1s = 1000) pro spuštění zvuku

iPercent : Pravděpodobnost spuštění zvuku

iVolumePercent : Pravděpodobnost hlasitosti spuštění zvuku

Zvuk je spuštěn po uběhnutí času mezi intervaly, poté je náhodně určena doba jeho následujícího spuštění (iTimeMin + rand() % (iTimeMax-iTimeMin)).

Příklad:

```
SoundRoom {
      sound001 { iID "500" iTimeMin "6000" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
      sound002 { iID "501" iTimeMin "3500" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
}
```

4.5 Definice objektů

Definice objektů přímo v definici levelu mohou využívat předem definované (**#define**) části shodné pro více objektů (textura, barva, odolnost...). Výskyt jména takové části vyvolá při kompilaci levelu vložení jejího těla na místo volání.

Příklad:

```
#define HODNOTA {  “100“ }
#define BGR { tex “50“ width “100“ }
define level {
   bgr002 { BGR heigth HODNOTA }
}
     …..kompilace
define level {
  bgr002 { tex “50“ width “100“  heigth “100“ }
}
```

4.5.1 Vlastnosti objektu

4.5.1.1 Základní vlastnosti

id : INT číslo objektu

tex : INT index textury z knihovny levelset, pro přístup do knihovny *.tex stačí zvýšit o 5000.

width : INT šířka obrázku

heigth : INT výška obrázku

repU : DOUBLE počet opakovaní textury do šířky, výchozí 1.0

repV : DOUBLE počet opakovaní textury do výšky, výchozí 1.0

shadow : BOOL vrhá stín = 1

hitSample : INT index zvuku při nárazu balónu (knihovna **sound.ini** a **soundroom**)

renderable : BOOL objekt je viditelný = 1

pos : VECTOR pozice na obrazovce [x, y] mezi [0..799, 0..599]

Hidden : INT objekt je skrytý do první kolize = 1

explode : INT příznak exploze objektu při kolizi = 1

explodeRadius : DOUBLE poloměr exploze

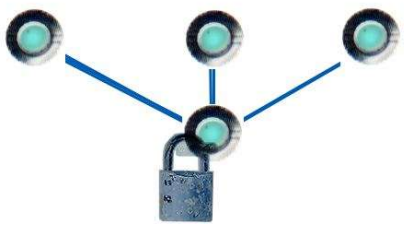
outrgba : COLOR barva objektu, výchozí RGBA "1.0, 1.0, 1.0, 1.0"

příklad poloprůhledné "1.0, 1.0, 1.0, 0.5"

4.5.1.2 Vlastnosti spojů

Spoje propojují jeden objekt k mnoha potomkům, u kterých jsou jeho prostřednictvím aktivovány / deaktivovány různé vlastnosti. Spojené objekty lze také řetězit a vyvolávat tak situace spínač otvírá dveře, tento spínač je ale zamčený (neaktivní) a pro otevření dveří je ho nutné aktivovat přes další spínač.

Teoreticky i když málo testovaná je možnost zamknout třemi spínači jeden spínač ve dvou úrovních. Pro jeho aktivaci by poté stačilo přepnout kterýkoliv ze tří spínačů v první úrovni.



Lze spojovat objekty typu v libovolných kombinacích :

spínač – spínače

spínač – cihly

spínač – generátory

spínač – pole

cihle – cihly

cihle – spínače

cihle – generátory

cihle – pole

pole – pole

renderLockLinks : BOOL příznak zobrazování spojů=1

lockListItems : INT počet vybíhajících spojů

lockListAction : INT funkce při aktivaci / deaktivaci spojů

1 … zamknout / odemknout

2 … odkrýt

lockList : INT ARRAY pole **id** objektů které se mají připojit

4.5.1.3 Vlastnosti klonování

clon : INT příznak klonování objektu

clonID : INT přírůstek k **id** klonů

clonW : INT počet klonů do šířky

clonH : INT počet klonů na výšku

clonAddTex : přírůstek k **tex** klonů

clonAddX : přírůstek mezi klony v ose x

clonAddY : přírůstek mezi klony v ose y

Příklad:

```
define level {
   wall01 { WALL_BASE clon "1" clonW "5" clonH "1" clonAddX "80.0, 0.0, 0.0" }
   alloc wall 5
   load wall wall01  //vytvoří 5 stejných kostek vedle sebe
}
```

4.5.1.4 Pohybové vlastnosti

move : INT typ pohybu

0 … bez pohybu

1 … pohyb po přiímce

2 … pohyb po kružnici

3 … skrolování

4 … konec levelu v případě že takový objekt opustí souřadnice [0..799, 0..599]

5 … náhodný pohyb

6 … náhodný pohyb po přiímkách

movieProp : INT vlastnosti pohybu při kolizích

0 … žádné

1 … konstantní

2 … převrácení vektoru rychlosti

3 … zastavení do další kolize

moveStop : INT počáteční nastavení pro zastavený pohyb = 1, kombinace s **movieProp = 3**

vVel : VECTOR vektor rychlosti pohybu

vMin : VECTOR levý horní roh kolizního obdélníku

vMax : VECTOR pravý dolní roh kolizního obdélníku

speed : DOUBLE rychlost

circlePos : VECTOR střed kružnicového pohybu

circleRadius : DOUBLE poloměr kružnicového pohybu

circleAlpha : DOUBLE počáteční natočení kružnicového pohybu

circleAlphaInc : DOUBLE přírůstek otáčení kružnicového pohybu

4.5.1.5 Pohybové rotační vlastnosti

rotate : INT příznak rotace objektu = 1

rotateBetaInc : DOUBLE přírůstek otáčení

rotateSpd : DOUBLE rychlost otáčení

rotateAlpha : DOUBLE počáteční natočení

4.5.1.6 Kolizní tvary

shapes : INT počet vlastních tvarů

test : INT příznak testu kolizí

0 … nedetekovat

1 … použít tvar vertex, výchozí

2 … použít definované tvary shapes

Tvarů lze mít libovolný počet a různě je kombinovat.

kolizní tvar **vertex** : základní tvar, množina vrcholů tvaru

Příklad: (4 vrcholy = 4 * VEKTOR, vytvoří tvar 50 * 20)

```
wall01 {
   vertex {
                v1 { v "  0.0,    0.0,   0.0"   }
                v2 { v " 50.0,    0.0,   0.0"   }
                v3 { v " 50.0,  20.0,   0.0"   }
                v4 { v "  0.0,  20.0,   0.0"   }
   }
}
```

kolizní tvar **intersect_sphere** : množina kolizních kruhů se středy a poloměry

Příklad: (jeden kruh s1, **test=2** použije tvar pro kolize a **shapes=1** ho alokuje v paměti)

```
wall02 {
              test     "2"      shapes "1"
              intersect_sphere {
                   s1 { radius "10.0"
                        o      "10.0, 10.0, 0.0"
                   }
               }
}
```

Kolizní tvar **intersect_poly** : množina kolizních polynomů s vektory AB.

Příklad: (jeden polynom shape 001, vektory ab01-06 s počátečním A a koncovým B)

```
wall02 {
            test    "2"     shapes "1"
             intersect_poly {
               shape001 {
                 ab06 { A "80.0,  16.0,   0.0"    B "0.0,  16.0,  0.0" }
                 ab05 { A "72.0,   4.0,   0.0"    B "80.0, 16.0,  0.0" }
                 ab04 { A "60.0,   0.0,   0.0"    B "72.0,  4.0,  0.0" }
                 ab03 { A "20.0,   0.0,   0.0"    B "60.0,  0.0,  0.0" }
                 ab02 { A "8.0,    4.0,   0.0"    B "20.0,  0.0,  0.0" }
                 ab01 { A "0.0,   16.0,   0.0"    B "8.0,   4.0,  0.0" }
               }
             }
}
```

4.5.2 Definice objektu pozadí

Pozadí bývá většinou statická textura, ale dají se mu přiřadit i pohybové funkce společné s dalšími objekty. S objekty tohoto typu neprobíhají žádné detekce kolizí a není tak nutné definovat jejich kolizní tvary.

4.5.3 Definice objektu zeď

Od zdí se odrážejí balóny. V každém levelu by mělo být alespoň několik zdí aby balón nevylétával z obrazu. Namísto zdí lze ale použít pálky. Stěny se mohou pohybovat, rotovat nebo využívat dalších funkcí společných všem objektům.

Proměnné pouze pro stěny :

net : BOOL stěna bude funkční a viditelná po sebrání efektu **NET**

net_switch : BOOL nastavit = **1** pro sítě

4.5.4 Definice objektu cihly

Balóny se odráží i od cihel, ale na rozdíl od zdí vydrží omezený počet zásahů, po kterém se roztříší. Cihly mohou mít různou velikost a tvar přičemž skóre za jejich zničení se vypočítá z jejich obsahu. Všechny typy pohybů a rotací mohou zvýšit obtížnost levelu.

Cihly mohou dále využívat vlastnosti spojů, které fungují následovně. Z jedné cihly s **id 1** vychází například čtyři zamíkací spoje do cihel s **id 2,3,4 a 5** tím budou 2,3,4,5 zamčené. Po zničení cihly **id 1** budou spoje odemčeny a až nyní lze zničit i 2,3,4,5.

```
Příklad :
brick001 { id "1" BRICK001  pos "195.0,  270.0,  0.0"  endurance "1.0"
                lockListItems "4"  lockList "2 3 4 5" }
brick002 { id "2" BRICK001  pos "195.0,  230.0,  0.0"  endurance "1.0" }
brick003 { id "3" BRICK001  pos "215.0,  230.0,  0.0"  endurance "1.0" }
brick004 { id "4" BRICK001  pos "230.0,  230.0,  0.0"  endurance "1.0" }
brick005 { id "5" BRICK001  pos "245.0,  230.0,  0.0"  endurance "1.0" }
```

Nebo odkrývající spoje. Z jedné cihly s **id 5** vychází dva odkrývající spoje do cihel s **id 3 a 4** tím budou 3,4 skryté. Po zničení cihly **id 5** budou spoje odemčeny, cihly 3 a 4 začnou být aktivní a lze je zničit.

```
Příklad :
brick001 { id "5" BRICK001  pos "195.0,  270.0,  0.0"  endurance "1.0"
                lockListItems "2"  lockList "3 4"  lockListAction "2" }
brick002 { id "3" BRICK001  pos "195.0,  230.0,  0.0"  endurance "1.0" }
brick003 { id "4" BRICK001  pos "215.0,  230.0,  0.0"  endurance "1.0" }
```

Proměnné pouze pro cihly :

bonus : INT index efektu, který by vypadl po zničení cihly (endurance <= 0)
lze použít i čísla skupin efektů :
100 … pro všechny možné
101 … efekty na balón
102 … efekty na level (globální efekty)
103 … efekty na pálku

bonusHard : INT při hodnotě = 1 bude mít efekt okamžitý účinek po rozbití cihly

percent : INT procenta výskytu bonusu v kostce, generování po nahrání levelu

endurance : DOUBLE odolnost kostky. Pro rozbití na jeden zásah = 1.0, na dva zásahy = 2.0. Základní balón bere při zásahu 1.1 z endurance.

vBonusVelocity : VEKTOR nepovinné určení počátečního vektoru rychlosti pro vypadávající bonus. Při neuvedení bude použit vektor z prostředí úrovně **LevelEnv**.

4.5.5 Definice objektu pálky

Pálka odráží balóny a ovládá se myší. Při kolizi s efektem získává další rozšíření jako jsou zbraně, změna velikosti, magnet apod. Pálka se pohybuje asynchronně s veškerým pohybem v úrovni, sleduje souřadnice myši a je tak možné ji přesunout z jednoho kraje na druhý během jednoho snímku. Záleží jen na reakci uživatele a citlivosti senzorů. Takový pohyb přináší problémy. Například když pálka vjede do balónu. Ovládat pálku je tak základní klíč k postupu do dalších úrovní.

Pálky se pohybují po vektorech v horizontálním nebo vertikálním směru a může jich být libovolný počet. Dále je možné mít různé tvary pálek s odlišnými obrázky. Balón je nasazen na první nahranou pálku.

left : DOUBLE levý okraj

right : DOUBLE pravý okraj

up : DOUBLE horní okraj

down : DOUBLE dolní okraj

initDir : VEKTOR směr natočení pálky a příjezdu nové pálky

Příklad : pálka u dolního kraje obrazu, pohyb horizontální

```
#define PAD_TEXTURE_NORMAL  { "200" }
#define PAD_DOWN {
            tex PAD_TEXTURE_NORMAL       shapes "1"  test    "2"
                hitSample "44"      initDir "0.0, 200.0, 0.0"
                width   "80.0"                  heigth  "20.0"
            vertex {
              v1 { v "  0.0,   0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
              v2 { v " 80.0,   0.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
              v3 { v " 80.0,  16.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
              v4 { v "  0.0,  16.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
            }
            intersect_poly {
              shape001 {
                l02 { A "4.0,    .0,   0.0"    B "80.0, 10.0,  0.0" }
                l01 { A "0.0,   10.0,   0.0"    B "40.0,  .0,  0.0" }
              }
            } //intersect_poly
          } //PAD_DOWN
define level {
  pad003 { PAD_DOWN   pos "0.0, 0.0, 0.0"  left "192.0" right "608.0" up "480.0" down "480.0" }
  alloc pad 1
   load pad pad003
}
```

4.5.6 Definice objektu spínač

Spínač mimo základních vlastností pracuje ve dvou stavech zapnuto/vypnuto. Stavy jsou přepínány kolizí s balónem nebo efektem **SWAP**. Dalším rozšířením je bonusový spínač, který ve stavu zapnuto přidává procenta ke každému skóre. Bonusové spínače lze vhodně řadit jako například 1x 2x 5x efektu u stolů her typu pinball.

switchTime : DOUBLE nastavení větší než 0.0 aktivuje časový spínač, který v dalším stavu po přepnutí vydrží jen určitou dobu (v sekundách)

switchOn : BOOL počáteční nastavení 1 = zapnuto

texOff : INT index textury knihovny **levelset** pro vypnutý spínač

addSwitchBonus : INT přidávání % ke skóre v zapnutém stavu

4.5.7 Definice objektu generátor

Generátor efektů má mnoho společného se spínači. Pokud je generátor zapnutý vyplivne při kolizi s balónem efekt a na nějaký čas se vypne.

switchTime : DOUBLE nastavení časové délky po kterou je generátor neaktivní po vyplivnutí (vygenerování) efektu (v sekundách)

switchOn : BOOL počáteční nastavení 1 = zapnuto

bonusHard : INT při hodnotě = 1 bude mít efekt okamžitý účínek

bonus : INT index efektu který vždy vypadne, viz **definice cihly**

vBonusVelocity : VEKTOR nepoviné určení počátečního vektoru rychlosti pro vypadávající bonus. Při neuvedení bude použit vektor z prostředí úrovně **LevelEnv**.

4.5.8 Definice objektu pole

Pole jsou aktivní pozadí, které mají při kolizi s balónem na něj vliv. Pole přebírají všechny základní vlastnosti včetně pohybů dvou polí se vzájemnou teleportací balónů mezi nimi.

fieldType : INT typ pole

0 … žádné

1 … na balón působí síla

2 … smrtící pole, při kolizi je balón ztracen stejně jako opustil obraz

3 … posouvací, při kolizi odsunou balón o vektor **fieldForce**

4 … teleportační, teleporují balón do jednoho z polí ze seznamu připojených objektů, u více jak jednoho pole se vybírá náhodně.

5 … odpuzující, odpuzují balón silou **fieldG** od středu **fieldForce** úměrnou jeho vzdálenosti od něj

6 … přitažlivá, přitahují balón silou ke středu

fieldG : DOUBLE multiplikátor působící síly

fieldForce : VEKTOR vektor působící síly

4.6 Nahrání objektů

alloc : alokace počtu objektů daného typu, přesná hodnota urychlí nahrávání,
bgr, field, wall, brick, switch, generator, pad
příklad : **alloc bgr 10** ….alokuje 10 pozadí

load : nahrání objektu daného typu
bgr, field, wall, brick, switch, generator, pad
příklad : **load bgr bgr001** ….nahraje pozadí bgr001

Příklad: (nahraje jedno pozadí)

```
define level {
      bgr001 { BGR001    pos "0.0,  0.0,  0.0" }
      alloc bgr 1
      load bgr bgr001
}
   … jiný způsob bez předalokace paměti
define level {
      bgr001 { BGR001    pos "0.0,  0.0,  0.0" }
      load bgr bgr001
}
```

4.7 Používání polí

Pro snadnější rozmisťování cihel (verze 0.4.2, později i ostatní objekty) lze použít pole. Pole se definují ve vnořeném adresáři **AddArray** definice levelu. Pole klonuje skupinu objektů a posouvá je o zadaný vektor včetně vlastností spojů. Definici každého pole, kterých může být libovolný počet tvoří číslo skupiny a adresář vektorů míst kam všude se má skupina naklonovat.

4.7.1 Definice pole

groupId : INT číslo skupiny
clonAddTex : INT zvýšení indexu textury všem klonům skupiny
clonID : INT zvýšení ID všem klonům skupiny
vectors : DIR vnořený adresář vektorů

4.7.2 Definice vektoru

vMove : VEKTOR vektor relativního posunutí
clonAddTex : INT zvýšení indexu textury všem klonům skupiny
clonID : INT zvýšení ID všem klonům skupiny
lockListAddID : INT zvýšení id v tabulce lock list, příznak klonování lock listu > 0

4.7.3 Příklad

Příklad: klonuje skupinu **1**, každému jejímu členu zvýší ID o **20**, celou skupinu klonuje dvakrát vektory **move001** a **move002**

```
define AddArray {
   array001 {   clonID "20" groupId "1"


            vectors {
              move001 { vMove "0.0, -100.0, 0.0" }
              move002 { vMove "200.0, -100.0, 0.0" }
            }
           }
} //AddArray
```

4.8 Hromadné přidání efektů

Namísto přiřazení efektu každé kostce zvlášť lze do levelu přidavát efekty globálně. Vnořený adresář **AddBonus** definice levelu obsahuje libovolný počet efektů, jejich počty a pravděpodobnosti. Po nahrání všch kostek jsou přiřiřazeny těm, které žádný bonus doposud nemají. Některé z efektů nemusí být přiřazeny v případě, že úroveň tvoří méně cihel než je počet přidaných efektů.

Definice :

bonus : INT typ efektu

iNumber : INT počet efektů na přidání

iPercent : INT pravděpodobnost přidání efektu (pro každý zvlašť)

iHard : INT okamžitý účinek viz definice cihly

Příklad: přidá 14 efektů čtyřech typů

```
#include "levels/bonus.ini"  // vloží definice efektů např  #define EFFECT_MAGNET    { bonus "4" }
define level {
   define AddBonus {
      bonus001 { EFFECT_MAGNET      iNumber "1"  iPercent "101" iHard "0" }
      bonus002 { EFFECT_BALL_FORK_2 iNumber "4"  iPercent "101" iHard "0" }
      bonus003 { EFFECT_BALL_BONUS  iNumber "3" iPercent "101" iHard "0" }
      bonus004 { EFFECT_BONUS       iNumber "6"  iPercent "101" iHard "0" }
      //  bonus005 { bonus  "4"    iNumber "1"  iPercent "101" iHard "0" } ekvivalentní s bonus001
   } //AddBonus
} // level
```

4.9 Příklad kompletní definice levelu

```
//--------------------------------------------------------------------------------
// level

#include "levels/pads.ini"
#include "levels/sound.ini"
#include "levels/bonus.ini"

#define BGR001 {
                tex "475"
                width  "800.0"    heigth     "600.0"
                vertex {
                 v1 { v "  0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                 v2 { v "800.0,    0.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                 v3 { v "800.0,  600.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                 v4 { v "  0.0,  600.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                }
               }

#define BGR002 {
                tex "489"
                width  "425.0"    heigth   "415.0"
                repU   "5.0"      repV     "5.0"
                vertex {
                 v1 { v "0.0,   0.0,     0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                 v2 { v "425.0, 0.0,     0.0"  UV "5.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                 v3 { v "425.0, 415.0,   0.0"  UV "5.0, 5.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                 v4 { v "0.0,   415.0,   0.0"  UV "0.0, 5.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
                }
               }
```

```
#define FIELD_G {
        width   "200.0"    heigth   "200.0"
        outrgba "1.0, 0.8, 0.5, 1.0"
        fieldType "4"
        shadow "0"   tex  "0"
        test     "2"      shapes "1"
        vertex {
         v1 { v "0.0,    0.0,     0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v2 { v "200.0,  0.0,     0.0"  UV "5.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v3 { v "200.0, 200.0,    0.0"  UV "5.0, 5.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v4 { v "0.0,   200.0,    0.0"  UV "0.0, 5.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
        }
        intersect_sphere {
            s1 { radius "100.0"
                 o      "100.0, 100.0, 0.0"
            }
        }
       }

#define WALL_UP {
        tex "471"  hitSample "158"
        width  "800.0"    heigth     "5.0"
        vertex {
         v1 { v "  0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v2 { v "800.0,    0.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v3 { v "800.0,    5.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v4 { v "  0.0,    5.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
        }
       }

#define NET_UP {
        tex "481"  hitSample "158"
        width  "470.0"    heigth     "20.0"
        repU   "14.0"     repV   "1.0"
        shadow "0"
        vertex {
         v1 { v "0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v2 { v "470.0,  0.0,   0.0"  UV "14.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v3 { v "470.0,  20.0,   0.0"  UV "14.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
         v4 { v "0.0,    20.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
        }
       }
```

```
#define WALL_LEFT {
              tex "471"  hitSample "158"
              width  "5.0"    heigth     "410.0"
              vertex {
               v1 { v "  0.0,   20.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v2 { v "  5.0,   20.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v3 { v "  5.0,  580.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v4 { v "  0.0,  580.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
              }
             }
#define WALL_TITLE_LEFT {
              tex "480"  hitSample "158"   width  "10.0"    heigth     "410.0"
              repU   "1.0"     repV       "10.0"
              vertex {
               v1 { v "  0.0,  0.0,     0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v2 { v " 10.0,  0.0,     0.0"  UV "1.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v3 { v " 10.0,  410.0,   0.0"  UV "1.0, 10.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v4 { v "  0.0,  410.0,   0.0"  UV "0.0, 10.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
              }
             }
#define WALL_TITLE_UP {
              tex "482"  hitSample "158"
              width  "416.0"    heigth     "10.0"
              repU   "5.0"      repV       "1.0"
              vertex {
               v1 { v "0.0,      0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v2 { v "416.0,    0.0,   0.0"  UV "4.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v3 { v "416.0,   10.0,   0.0"  UV "4.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v4 { v "0.0,     10.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
              }
             }
#define WALL_TITLE_RUP {
              tex "484"  hitSample "158"
              width  "10.0"    heigth     "10.0"
              vertex {
               v1 { v "0.0,     0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v2 { v "10.0,    0.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v3 { v "10.0,   10.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
               v4 { v "0.0,    10.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
              }
             }
```

```
#define WALL_TITLE_LUP {
            tex "483"  hitSample "158"
            width  "10.0"    heigth     "10.0"
            vertex {
             v1 { v "0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
             v2 { v "10.0,    0.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
             v3 { v "10.0,   10.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
             v4 { v "0.0,    10.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
            }
           }
#define BRICK001 {
            tex "474"  hitSample "167"
            width  "50.0"    heigth     "20.0"
            vertex {
             v1 { v "  0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
             v2 { v " 50.0,    0.0,   0.0"  UV "0.5, 0.0, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
             v3 { v " 50.0,  20.0,   0.0"  UV "0.5, 0.25, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
             v4 { v "  0.0,  20.0,   0.0"  UV "0.0, 0.25, 0.0"  rgba "1.0, 1.0, 1.0, 1.0" }
            }
           }
#define BRICK002 {
            tex "474"  hitSample "167"
            width  "50.0"    heigth     "20.0"
            vertex {
             v1 { v "  0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
             v2 { v " 50.0,    0.0,   0.0"  UV "0.5, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
             v3 { v " 50.0,  20.0,   0.0"  UV "0.5, 0.25, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
             v4 { v "  0.0,  20.0,   0.0"  UV "0.0, 0.25, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
            }
           }

#define BRICK_BOMB {
            tex "528"  hitSample "167"
            width  "20.0"    heigth     "20.0"
            explode "1"  explodeRadius "100.0"
            vertex {
             v1 { v "  0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
             v2 { v " 20.0,    0.0,   0.0"  UV "0.5, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
             v3 { v " 20.0,  20.0,   0.0"  UV "0.5, 0.25, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
             v4 { v "  0.0,  20.0,   0.0"  UV "0.0, 0.25, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
            }
           }
```

```
#define SWITCH_001 {
                shadow     "0"       test     "2"      shapes "1"  hitSample "157"
                switchTime "0.0"     switchOn "1"
                width      "20.0"    heigth   "20.0"

               vertex {
                v1 { v "  0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
                v2 { v " 20.0,    0.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
                v3 { v " 20.0,   20.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
                v4 { v "  0.0,   20.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
               }

               intersect_sphere {
                  s1 { radius "10.0"
                     o      "10.0, 10.0, 0.0"
                  }
               }
              }

#define GEN_001 {
                     test     "2"      shapes "1"  hitSample "157"
                switchTime "5000.0"     switchOn "0"
                width      "40.0"    heigth   "40.0"
               vertex {
                v1 { v "  0.0,    0.0,   0.0"  UV "0.0, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
                v2 { v " 40.0,    0.0,   0.0"  UV "1.0, 0.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
                v3 { v " 40.0,   40.0,   0.0"  UV "1.0, 1.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
                v4 { v "  0.0,   40.0,   0.0"  UV "0.0, 1.0, 0.0"  rgba "0.5, 1.0, 1.0, 1.0" }
               }
               intersect_sphere {
                  s1 { radius "20.0"
                     o      "20.0, 20.0, 0.0"
                  }
               }
              }
```

```
define level {
   id       "1"
   levelLoadingText "your first level \enjoy"
   nextLevelAddId   "1"
   loadingImg       "1"
   loadingText      "first level"
   forkImg1         "2"
   forkImg2         "3"
   forkText1        "Level 2"
   forkText2        "Level 3"
   fork1AddId       "1"
   fork2AddId       "2"
   gravitation      "0.0,   0.0,   0.0"
   env_velocity     "0.0,  10.0,   0.0"
   env_speed        "1.0"
   env_mass         "1.0"
   nextBonusLevel   "0"
      bonusLevel    "0"
   levelBgrEnv      "1"

   SoundRoom {
      sound001 { iID "500" iTimeMin "6000" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
      sound002 { iID "501" iTimeMin "3500" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
      sound003 { iID "502" iTimeMin "1500" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
      sound004 { iID "503" iTimeMin "1000" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
      sound005 { iID "504" iTimeMin "5000" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
      sound006 { iID "505" iTimeMin "2500" iTimeMax "10000" iPercent "50"  iVolumePercent "100" }
   }

   define AddBonus {
      bonus002 { EFFECT_BALL_FORK_2 iNumber "6"  iPercent "101" iHard "0" }
      bonus003 { EFFECT_BALL_BONUS  iNumber "6" iPercent "101" iHard "0" }
   } //AddBonus

   define AddArray {
      array001 {   clonID "20" groupId "1"
               vectors {
                  move001 { vMove "0.0, -100.0, 0.0" }
                  move002 { vMove "200.0, -100.0, 0.0" }
               }
              }
   } //AddArray
```

```
bgr001 { BGR001    pos "0.0,  0.0,  0.0" }
bgr002 { BGR002    pos "187.0,  123.0,  0.0"  }
alloc bgr 2
load bgr bgr001
load bgr bgr002



alloc field 1
field_g01 {  FIELD_G    pos "300.0,  200.0,  0.0"  renderable "1"
           fieldForce  "0.0,   0.0,     0.0"  fieldG "36.0" }
load field field_g01

wall004 { WALL_UP    pos "0.0,  123.0,  0.0"  renderable "0" }
wall005 { NET_UP     pos "180.0,  530.0,  0.0"  renderable "0"  net "1"  net_switch "1" }
wall006 { WALL_LEFT  pos "187.0, 0.0,  0.0"   renderable "0" }
wall007 { WALL_LEFT  pos "608.0, 0.0,  0.0"   renderable "0" }
wall009 { WALL_TITLE_LEFT  pos "182.0, 128.0,  0.0"   renderable "1" }
wall010 { WALL_TITLE_UP    pos "192.0, 118.0,  0.0"  renderable "1" }
wall011 { WALL_TITLE_LUP   pos "182.0,  118.0,  0.0"  renderable "1" }
wall012 { WALL_TITLE_RUP   pos "608.0,  118.0,  0.0"  renderable "1" }
wall008 { WALL_TITLE_LEFT  pos "608.0, 128.0,  0.0"   renderable "1" }
alloc wall 9
load wall wall004
load wall wall005
load wall wall006
load wall wall007
load wall wall009
load wall wall010
load wall wall011
load wall wall012
load wall wall008

brick001 { id "20" groupId "1" BRICK001  pos "220.0,  270.0,  0.0"  endurance "1.0"  }
brick002 { id "21" groupId "1" BRICK001  pos "280.0,  270.0,  0.0"  endurance "1.0"  }
brick003 { id "22" BRICK001  pos "340.0,  270.0,  0.0"  endurance "1.0"  }
brick004 { id "23" BRICK001  pos "400.0,  270.0,  0.0"  endurance "1.0"  }
brick005 { id "24" BRICK001  pos "460.0,  270.0,  0.0"  endurance "1.0"  }
brick006 { id "25" BRICK001  pos "520.0,  270.0,  0.0"  endurance "1.0"  }
brick007 { BRICK002  pos "220.0,  300.0,  0.0"  endurance "1.0"  }
brick008 { BRICK002  pos "280.0,  300.0,  0.0"  endurance "1.0"  }
```

```
  brick009 { BRICK002  pos "340.0,  300.0,  0.0"  endurance "1.0"  }
  brick010 { BRICK002  pos "400.0,  300.0,  0.0"  endurance "1.0"  }
  brick011 { BRICK002  pos "460.0,  300.0,  0.0"  endurance "1.0"  }
  brick012 { BRICK002  pos "520.0,  300.0,  0.0"  endurance "1.0"  }
  brick013 { BRICK_BOMB  pos "390.0,  350.0,  0.0"  endurance "1.0" bonus "100" percent "0" }
  alloc brick 13
  load brick brick001
  load brick brick002
  load brick brick003
  load brick brick004
  load brick brick005
  load brick brick006
  load brick brick007
  load brick brick008
  load brick brick009
  load brick brick010
  load brick brick011
  load brick brick012
  load brick brick013

  switch_001 { SWITCH_001  pos "400.0, 350.0,  0.0"   renderable "1"  id "50"
            lockListItems "6" lockList "20 21 22 23 24 25" }
  alloc switch 1
  load switch switch_001

  gen_001 { GEN_001  pos "400.0, 350.0,  0.0"   renderable "1"  bonus "4"
              moveProp "3"   moveStop "1"
              move  "1"
              vMin "300.0  350.0  0.0"  vMax "500.0  350.0  0.0"  vVel "10.0  0.0  0.0"
              speed "4.0"
        }
  alloc generator 1
  load generator gen_001

  pad003 { PAD_DOWN   pos "0.0, 0.0, 0.0"  left "192.0" right "608.0" up "480.0" down "480.0" }
  alloc pad 1
  load pad pad003
}
//-------------------------------------------------------------------------
// e n d
```