**Project 7&8**

**Brad the Brat**

Design Document

*An Egocentric Game*

Game Boy Advance

Bruno Jordaan

Phan Vo Bang

Joeri Levefre

Niels Keetels

Peter Nauta

*Project Group GDD2b1*

**Colophon**

Bruno Jordaan (content)

Peter Nauta (content / planning)

Phan Vo Bang (concept art)

Joeri Levefre (concept art)

Niels Keetels (technical)

**Index**

1. Defining the Game Page 4

2. Core Gameplay Page 5

2.1 The Main Game View

2.2 Core Player Activity

2.3 Controls

2.4 In-game User Interface

3. Contextual Gameplay Page 8

3.1 Shell Menu’s

3.2 The Nuts and Bolts

4. Detailing the Game Page 10

4.1 Who is Brad?

4.2 Level, Mission and Area Design

4.3 Method of Interaction

4.4 Effect of Interaction

5 Covering the Assets Page 16

5.1 Sprites and Animations

5.2 Music Assets

5.3 Sound Effects Assets

Appendix I: GBA Technical Specifications Page 21

**1. Defining the Game**

Brad the Brat is, as the title implies, (all) about Brad.

“*Lots of other kids were coming in, which was easily explained since he was present. Two of them had even been early to see him enter the kindergarten. He had dropped his coat on the floor and one of the teachers had picked it up and hung it at the door. He hadn’t expected less of her. The sun was shining in. When waking up this morning he wanted it to be a beautiful day, so it had to be a beautiful day.*

*Kids were walking around him. One small Asian boy was running around in a superhero suit and constantly bumping into chairs, tables and other kids. Then there was this bald boy who was bragging about everything he had achieved yesterday, in some computer game. Teachers were furiously looking for a girl that was supposed to be here, but kept disappearing out of sight. Breakfast had been pretty bad; although he already was an accomplished cook himself, he preferred to let his mother make it for him. If she was spending her time, then best spend it on him.*

*He called for the shaking kid in the corner to get him one of the toys from the corner. The kid looked up from his drawing, which he named ‘game engine’, walked to the fire-truck and brought it to him. A fourth kid was furiously pointing at him, but then turned around to find the skinny boy who had taken away his guitar. Good, he was going to like it here. There were enough others around to do his bidding. After all, if not his, whose bidding would they do?”*

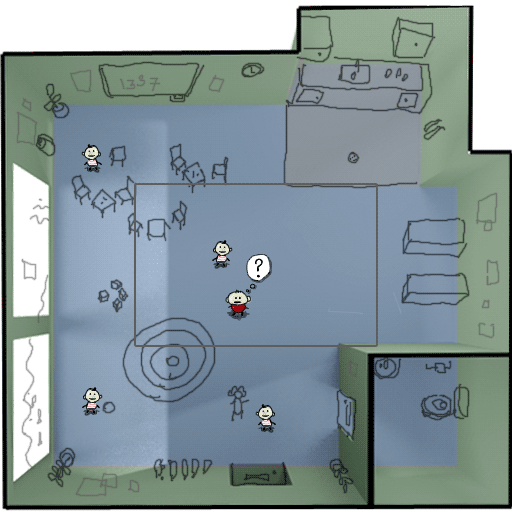
Brad is a demanding, ego-centric child in kindergarten, the playing area of the game. The goal of the player is to keep Brad happy during the course of a single day. This can be accomplished by fulfilling Brad’s primary needs, by preventing him from getting bored, by giving him attention and recognition. However, doing so is never easy, since none of the other kids in kindergarten like Brad very much, especially not when he is bragging, pouting or complaining. The game is about managing Brad’s environment so it becomes possible to get through that single day.

**2 Core Gameplay**

**2.1 The Main Game View**

The main view in Brad the Brat is top-down. The environment is (fake) 3d, but the sprites in the game are all in 2d. All levels take place in the same kindergarten environment that covers six full Game Boy screens. Players can slide over the level fluently, since all the information is stored in the Game Boy’s memory.

Within our level, the player will be presented with the boss Brad, other kids, teachers, pets, toys and other objects with which he/she can interact. Some of these have their own will and move around on their own accord.



**2.2 Core Player Activity**

The core gameplay activity for the player is to keep Brad’s ego high, for one full day, by interacting with the objects around him. These objects can be other kids, teachers, toys, consumables etc.

Brad’s ego can be monitored by looking at his Egobar. Four variables make up this Egobar:

* Needs (example: food, drinks)
* Activity (example: playing, moving around)
* Attention (example: talking, doing activities together)
* Recognition (example: stimulating his super-ego)

These variables need to be balanced: so although you can’t starve Brad, you also can’t over-feed him (since he will feel sick).

The players can’t control Brad directly, so in order to keep Brad’s needs balanced, they will need to interact with the environment around him.

The different objects in his environment can be picked up, dropped or turned on/off. The characters can be given suggestions what to do in order to satisfy Brad’s needs. Depending on their attitude towards Brad, they will decide whether or not they will actually give in to this suggestion.

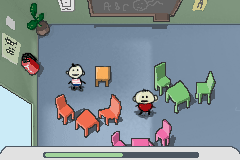
**2.3 Controls**

Below are the proposed controls used in *Brad the Brat*.

* **D-pad:** Moving the cursor around. When the cursor moves over an object that can be interacted with, the ‘item’ will become highlighted. When the cursor comes near the edge of the screen, it will scroll accordingly. When objects are picked up by the player, the D-pad is used to move the object around.
* **A-key**: going a level deeper in a menu. Pick up an item. Select the highlighted object to interact with it.
* **B-key**: going up a level in a menu. Drop an item.
* **L-key**: Hard select. Instead of having to move the cursor, the cursor will go to the next kid in the visible area of the entire playground.
* **R-key**: Hard select. Instead of having to move the cursor, the cursor will go to the previous kid in the visible area of the entire playground.
* **Select**: Centre on Brad
* **Start**: Game pauses. Show screen with detailed Egobar breakdown. From here, the player can use L/R-keys to switch between the screen with the Egobar breakdown and the screen with the briefing. Pressing start again will un-pause the game.

**2.4 In-game User Interface**

Since the screen of the Game Boy advance is quite small, there will be a minimum of interface elements on the main playing screen. The only thing visible for the player during play is the Egobar on the bottom of the screen and its state at that moment.

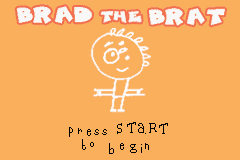


More information on the variables that together constitute the Egobar can be seen by pressing the **start** button at any time. A menu will appear where the player can see the state of all the individual variables that make up Brad’s ego.

**3 Contextual Gameplay**

**3.1 Shell Menus**

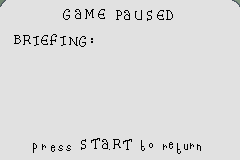
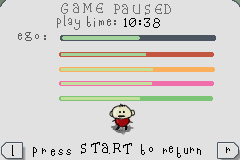
When starting the game, the player will be presented with the intro screen. This will contain the name of the game and the fact that the player has to press start in order to begin. Two iterations have been made for this screen.

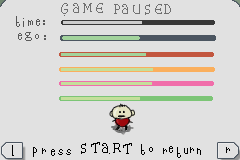
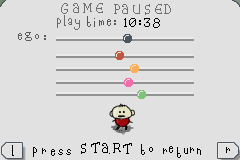


Upon pressing the **start** button, the game will pause and the player will be presented with a menu that gives more insight in the state of the variables that make up Brad’s Egobar, the so-called ‘ego breakdown screen’. In this screen the player can also see for how long he/she has been playing.

By using the **L-key** or **R-key**, the player can move to the ‘briefing screen’. This will show him/her what this level is about and what he/she has to achieve within the time limit. Pressing the **L-key** or the **R-key** will take him/her back to the other screen and pressing **start** again will take the player back to the game.

Several variants have been made for the ego breakdown screen.





**3.2 The Nuts and Bolts**

The playing environment consists of a 512x512 pixel grid. All the sprites used are 32x32 pixels. All kids, teachers and pets start the game with a neutral attitude towards Brad. Also, all of the variables that make up Brad’s Egobar in balance, and therefore his ego is perfect at the start of the game.

One day in the game consists of 8 hours and every 15 minutes in the game takes 30 seconds in real life. This means that the player has to keep Brad’s ego satisfied for 16 minutes.

The different variables that make up Brad’s ego, all drop during play. The speed in which these drop (from fast to slow):

* Activity
* Needs
* Attention
* Recognition

In total there will be 24 objects for the player to interact with:

* 1 Brad
* 4 static items
* 3 teachers
* 4 toys
* 2 pets
* 8 kids
* 2 consumable makers

If the player uses a ‘sentient’ entity (such as another kid, or a pet) to please Brad, it affects the attitude of that entity towards Brad. Entities with a very negative attitude towards Brad are unlikely to respond to the suggestions of the player. Given ‘rest’ (meaning: not being near Brad, and not having to do things for him), the attitude of the entities grows back to a neutral state. This is an exponential scale: it takes four times as long to get a 100% kid back to neutral as a kid who is only 50% annoyed1.

**4 Detailing the game**

**4.1 Who is Brad?**

Brad is 10 years old, and is used to getting what he wants, when he wants. His parents are filthy rich and very influential in the community, and they have always made sure nothing is lacking for him. In recent times, however, they took over most of businesses in town (including the kindergarten), and have been busy ‘milking it dry’. That’s why Brad spends his days in the kindergarten at the moment. This is also the reason the player ‘loses’ when he fails to keep Brad happy for a day – as the parents will come in, get Brad, and subsequently fire the teachers.

**4.2 Level, Mission and Area Design**

Since we are designing only a boss-encounter, in our case the last level of the game, we will give a general idea of the game as a whole, but focus mostly on the Brad-encounter.

The game is divided into different levels. In each level you will be presented with a kid, who has a specific (troublesome) character trait. It is up to the player to let the kid ‘survive’ in the kindergarten environment for one day. When this is a success, the player will go to the next level and is presented a new kid with a different character trait. The nine kids in the game form a constant – they are always there, just the kid who you have to keep happy changes. Brad is actually the ninth kid, but to give a better idea of how the previous levels are we have made a list and named the other ‘problem kids’ and teachers you have to deal with in the game:

* Bragging Bruno
* Phantastic Phan
* Nervous Niels
* Jealous Joeri
* Pointing Peter
* Seducing Sandra
* Missing Merel
* Manly Martijn
* Ever-smiling Evert (teacher)
* Anxious Anne (teacher)
* Deviant Dop (teacher)

All these kids have to be kept ‘satisfied’ for one day and all have different variables that make up their own ‘bar’ (in Brad’s case this is the Egobar).

Graphical outline: All objects in the game have an outline, a shadow and are filled with three color values. The color scheme that is used consists of secondary colors. The game perspective is 'bendable': though all objects have a certain form, their representation is up to the artist. The graphical style itself is childish.

Each kid has his own theme music that is playing in the background and which is connected to the ‘problem’ it poses to the player.

**4.3 Method of Interaction**

This is an attempt to classify the objects or entities within the game environment.

|  |  |  |
| --- | --- | --- |
| **Object / Entity** | **Class** | **Control** |
| Brad the Brat | Special | None |
| Kids | Dynamic | Indirect (suggestion) |
| Teachers | Dynamic | Indirect (suggestion) |
| Pets | Dynamic | Direct\* |
| Food & Drinks | Consumable | None |
| Toys | Moveable | Direct |
| Books / Clay / Paint | Moveable | Direct |
| TV | Static | Direct |
| Furniture | Static | Direct |

\*Although pets can be moved and manipulated directly, they can still get annoyed, bored, or angry, which causes them to wander of again as soon as they are ‘released’.

***Object / Entity Class***

* **Special**

Brad makes up a separate class by himself. His behavior, however, is similar to those of dynamic entities.

* **Dynamic**

Dynamic entities have ‘their own will’, meaning they move and act on their own accord, when the player is not interacting with them.

* **Consumable**

Consumable objects can be created and destroyed, but are otherwise similar to moveable objects.

* **Moveable**

Moveable objects can be picked up, moved around, and manipulated by the player, or by a (player-controlled) entity.

* **Static**

Static objects can be manipulated, but not picked up or moved around by the player, or by a (player-controlled) entity.

***Control***

* **None**

The player has no control, neither directly or indirectly, on the behavior of this entity.

* **Indirect (Suggestion)**

The player does not have full control over these entities, but can guide their behavior through ‘suggestions’ instead. Whether or not the entity behaves as the player wants is dependant on his ‘attitude’ towards the requested action, and towards other entities involved (mainly Brad).

* **Direct**

The player can directly manipulate these objects.

***Interacting with kids and teachers (dynamics)***

When hovering over a kid that the player can interact with, this kid will become highlighted. When using the A-button to select the kid, a menu with actions will appear. All the actions that the player can undertake are visually represented and appear within circles. With the R-button one can turn the actions clockwise; With the L-button one can turn the actions counter-clockwise.

The Egobar is replaced by an information-bar, which will give the player some information about the selected action.

The action that is selected to be performed is on top of the character and marked through showing it in a green circle. The action can be undertaken by pressing the A-button.

***Interacting with pets (dynamics)***

A pet is basically a toy that has his own will: during the game it will move around. When hovering over a pet, the pet will be highlighted and by pressing the A-button, a pet can be picked up. The pet can then be moved and dropped by pressing the B-button, or given to a kid to a kid by targeting that kid and pressing the A-button.

***Interacting with static objects***

Static objects can be turned on or off. They will have an effect on Brad wherever he is in the room and have an effect on other kids that are in the radius of the static object. The effect radius of effect is for all the static objects the same, the only difference is their appearance and the placement. There will be one static object in every corner.

***Interacting with movables***

When hovering over a toy, the toy will be highlighted and by pressing the A-button, a toy can be picked up. The toy can then be moved and dropped by pressing the B-button, or given to a kid by targeting that kid and pressing the A-button.

***Interacting with consumables***

Consumables are created by teachers, and have a short lifespan. Right after creation they are given to a kid (or Brad) and are consumed.

***Performed actions of the dynamic entities***

* Kids interacting with:
  + Self: walk around
  + Other kids: talk to other kid, play together with other kid
  + Toys: play with toy
  + Pets: play with pet
  + Food/Drinks: eat food, drink
  + Teachers: talk to teacher
* Teachers interacting with:
  + Self
  + Food/Drinks
  + Kids
* Pets interacting with:
  + Self
  + Kids
  + Other Pets

**4.4 Effect of Interaction**

Each action that is undertaken by the player to help improve Brad’s ego, has its effect on two variables. A problem arises when the player tries to increase one value: another value increases as well. This prevents the player from stimulating Brad in a mindless fashion and encourages smart play and planning.

For example: when Brad’s ‘activity bar’ is imbalanced, he needs something to do. If the player suggests another kid to play with him, not only his activity will go up, but also the attention bar will increase. Because of this, it is possible to imbalance one of the variables, while (trying to) balance another.

Whenever the player undertakes an action that fulfils Brad’s needs, a small + sign will appear over his head and then fade out. Around Brad there is an invisible circle which represents his influence range. Whenever Brad receives a +, all the other kids and teachers within this range, will receive a – and thus changing their attitude towards Brad.

We have introduced the *static objects* (4.3 Method of interaction) to give the players a way of moving Brad and his influence range around. Players will therefore be able to rotate between kids and teachers, giving them better control over whose attitudes they will affect with their actions.

Both kids and teachers that are interacting with Brad, have an attitude towards him. These start off neutral (all are over very open to Brad in the beginning of the level), but by using them as a means to please Brad, they will gradually turn negative, since they become annoyed with Brad’s way of using them and others. Influence on attitude will happen in two ways: when they are the actual means or when they are close enough to Brad (which is: in his influence range) to witness how he uses others as a means to achieve a goal.

When their attitude is bad, they will simply refuse to do something for Brad and the person has to be left alone for a while, before they can be used again. This attitude will change from bad into neutral again over time (see 3.2 The Nuts and Bolts for a more detailed description).

Feedback on their attitude is provided by: kids, teachers and pets. When hovering over a kid, teacher or pet, a thought balloon will show up with their attitude towards Brad at that moment.

Since Brad is more important than the other characters in the level, he will give more active hints of how he is doing. When one or more of the variables that make up his ego becomes seriously imbalanced, he will alert the player through an animation. By pressing the **start** button, the player will receive more information about the imbalance.



**5 Covering the Assets**

**5.1 Sprites and Animations**

All animations will consist of 2 or 3 sprites.

*Level*

* Single 512x512 Background Layer (6 screens)
* Introduction screen
* Start menu: ‘Ego Breakdown screen’
* Start menu: ‘Briefing-screen’
* The attitude ‘+’ sign
* The attitude ‘-’ sign
* Props [not yet defined]
  + Furniture
  + Fire Extinguisher
  + Plants
  + Sink
  + Water Machine
  + Fridge
  + Clock (With animation?)
* ‘Static’ Interact-able Objects
  + Television Set [x1]
  + School Board [x1]
  + Aquarium [x1]
  + Toys Corner [x1]

*Characters*

* Brad the Brat [x1]
  + Idle Animation
  + Walk Animation
  + Sitting Animation
  + Eating Animation
  + Drinking Animation
  + TV Hypnotize Animation
  + Watch TV Animation
  + Talk Animation
  + Play with Toy Animation (1 per toy)
  + Play with Pet Animation (1 per pet)
  + Get-item Animation
  + Drawing Animation
  + Crying Animation
  + Bored Animation
  + Bragging Animation
  + Imbalance Animation (‘alert’)
* Kids [x8]
  + Idle Animation
  + Walk Animation
  + Sitting Animation
  + Eating Animation
  + Drinking Animation
  + TV Hypnotize Animation
  + Watch TV Animation
  + Talk Animation
  + Listen Animation
  + Play with Toy Animation (1 per toy)
  + Play with Pet Animation (1 per pet)
  + Get-item Animation
  + Give-to Animation
  + Drawing Animation
  + Crying Animation
  + Being Annoyed with Brad Animation
* Teachers [x3]
  + Idle Animation
  + Walk Animation
  + Sitting Animation
  + Talk Animation
  + Listen Animation
  + Give Animation
  + Compliment Animation
  + Create Food Animation
  + Create Drink Animation
* Pets [x2]
  + Idle Animation
  + Walk Animation
  + Sit Animation
  + Sleep Animation

*Moveable Objects*

* Toys [x4]
* Food [1]
* Drink [1]

*Alphabet*

It turned out that on the Game Boy Advance there is no alphabet that can be accessed through hardware functions, so we were forced to either find an alphabet or design one ourselves.

At this stage of our preparations we have decided on the following alphabet to be used in our game.



The reason for this is that it has a childish look and it is still very well readable. Note that this alphabet is already being used on our first iterations of the interface and menu screens.

**5.2 Music Assets**

The music will be chip-tune and because of the tight budget of 256k for our game, 20k will be spend on music. The music will be converted to \*.mod format.

Brad’s Theme I 2-3mins

**5.3 Sound Effects Assets**

The sound effects will be in \*.wav format and we have a budget of 10k to spend on them. The sounds will be made in 22 kHz or 11 kHz, dependant on the quality of the last option. Sounds will be made for the following actions:

* Moving with L/R-buttons between menu’s
* Moving from one command/suggestion to another
* Acknowledge (as function for the A-button)
* Cancel (as function for the B-button)
* Acknowledge from kid / “I will follow the suggestion” / “hm-hm”
* Denial from kid / “I will not follow the suggestion” / “uh-uh”
* Brad eating sound
* Brad drinking sound
* Brad talking sound
* Brad superiority sound / music

**Appendix I: Game Boy Advance Technical Specifications**

*by Niels Keetels*

# Nintendo Game Boy Advance

16.78 MHz 32-bit ARM7tdmi RISC processor

96k VRAM (frame buffer)

32k IW-RAM, 256k EW-RAM (general purpose RAM)

Display resolution: 240x160 pixels

Refresh rate: 59.74 Hz

# Tile modes

Tile-modes enable us to use up to four scrolling and blendable background-layers of which some support hardware rotation and scaling features. These layers use a common 8-bit palette consisting of 256 color-indices. The first color in the palette is considered the transparent color. Background layers consist of 8x8 tiles in memory but it’s quite easy to support 16x16 tiles, we can have a tool to split them up in 8x8 tiles.

There are three different tile modes for Game Boy Advance. The number of available layers depends on the mode you’re using. Let’s have a look at the tile modes.

Mode 0

Mode 0 features four background layers which support hardware scrolling and transparency. None of these can be scaled and/or rotated though. The backgrounds can be up to 512x512 pixels large (16 x 16 tiles)

Mode 1

Mode 1 features three background layers of which one (BG2) supports rotation and scaling. BG2 can be up to 1024x1024 pixels (32 x 32 tiles). All backgrounds support scrolling and transparency too.

Mode 2

Mode 2 supports only two background layers but these can both be scaled and rotated. Both modes support scrolling and transparency too.

# Sprites

The Game Boy Advance is able to render up to 128 sprites simultaneously. Each sprite supports transparency and can be scaled, blended and rotated on hardware. Sprites use a common 256-color palette, just like backgrounds do. The first color-index in the palette is the transparency color. You can give sprites a priority flag so the can be rendered on top of layer 1 but underneath layer 2 for example.

We can use the following sprite sizes:

8x8 8x16 16x8

16x16 8x32 32x8

32x32 16x32 32x16

64x64 32x64 64x32

We have 32k of sprite-memory available, so it’s not possible to have 128 sprites of size 64x64. You can have 512 8x8 sprites in memory or 128 16x16 sprites for instance. Not all sprites have to be the same size of course.

If we use a tile mode with sprites we can have up to 510 colors on screen simultaneously since sprite and background palettes are separate.

What I’ve illustrated here only counts for tile-modes. In bitmap mode we can use sprites too however the sprite memory overlaps with frame buffer memory (VRAM is only 96k), that’s why we could only render a limited number of sprites in bitmap mode. For this project I assume we’ll be using a tile mode anyway.

# Bitmap modes

In bitmap mode we have access to the frame buffer (we can plot individual pixels), in contrast to tile modes where we can only render tiles and sprites. In general bitmap modes are quite slow because we have to implement the rendering in software. The pro of bitmap modes is that you can write your own visual effects and it’s easier to render pictures. By the way, it’s perfectly possible to switch instantly between a tile mode and a bitmap mode (i.e. for splash screen and game menu). There are three bitmap modes of which two support up to 32768 colors simultaneously (16-bit).

Mode 3

This mode is a real 16-bit mode but it’s quite slow and practically unusable for real-time graphics.

Mode 4

Mode 4 is the only bitmap mode which doesn’t support 16-bit graphics. It’s just plain palletized 8-bit again but this mode supports double-buffering. This feature makes animations look smoother since you can’t see the vertical retrace of the LCD screen.

Mode 5

This mode combines the best of the other two bitmap modes. It’s a 16-bit mode and it supports double buffering, however this comes at a cost: the screen is only 160x128 pixels large in this mode since there isn’t enough VRAM available to store 2 x 240x160 x 2 bytes.

# Sound

There are freeware MOD players available for Game Boy Advance. The music track can have up to four channels and the player only supports 8-bit samples. This does sound quite okay on speakers though.

# ROM

Theoretically we have as much space to store graphics and music as the size of the cartridge but my cartridge is only 16mb. Most early GBA games were 4mb in size. We have another option to put the rom on our GBA’s though, and that option is multiboot. For multiboot the ROM can only be 256k in size, but then it’s possible to send a ROM from your PC to the external working RAM of the Game Boy Advance (I’ve made a custom cable for that). We won’t need a cartridge that way.

# For this project

For project 7/8 I propose using mode 0 since we can guarantee 60 frames per second using hardware rendering. Tile-modes are more elegant to work with than bitmap modes anyway (and it’s way more challenging). I think our game is pretty doable in 256k so let’s have that as our requirement.

# File formats

I want sprites for every moving object or character in 8-bit BMP format. For animations I want the frames vertically aligned (i.e. a 16x16 sprite animation of 3 frames becomes a bitmap of 16x48 in size). A walking animation only needs to be 3 frames for each direction (up, down, left and right). Each frame counts as a sprite in memory (32k) but not as an OAM entry (OAM entries are sprites that get rendered on screen, max 128). Static objects which can’t be moved or picked up don’t belong in sprites, we use backgrounds for that. We use sprites only for moving/dynamic objects and characters.

The palette is something that needs caution, it might be a good idea to create the palette with a rainbow gradient or something in Photoshop and save that as an 8-bit BMP file to work with (don’t forget to convert it to an 8-bit palette by clicking image->mode->indexed mode). Every member of our team who’s involved with the graphics needs to use the same palette. Remember that the first color-index in the palette is reserved for transparency!

The tiles for backgrounds can all be saved in one big BMP file (also 8-bit and please mind the transparency color). I’m not sure which tool we’ll use to design the backgrounds yet but at least we could have some tiles already.

The music (which I’ll probably be making myself) needs to be in MOD format. Up to 16 mono samples (8-bit) can be used in a maximum of 4 channels. The player supports stereo sound.

# Summary

# Backgrounds

We’re using mode 0, so we have up to 4 layers available (128x128, 256x256 or 512x512 pixels). We’ll probably be using just one or two backgrounds but I don’t see a need for rotation and scaling. For these layers a set of tiles needs to be created, I don’t care whether they’re 8x8, 16x16 or 32x32, just go with what looks good on a 240x160 display. If you save all the tiles in one BMP file you won’t have any problems with merging palettes later. The common transparency color is R: 128 G: 0 B: 128 but you can use any color for transparency (has to be the first color in the palette!).

# Sprites

For characters and dynamic/moving objects we use sprites. We can have up to 128 sprites simultaneously on screen. Choose a fixed sprite-size before you start a new animation (if your character is more than 16 pixels high, then have each frame of his animation size16x32). All sprites use a common sprite-palette and please remember the first color-index being transparent.

# Sprite animations

For animations you are not to use more frames than needed. For a walking animation three frames for each direction (left, right, up, down) is sufficient. Make sure you’re doing the frames of each direction in the same order (for instance, left-foot-in-front, both-feet-in-the-middle, right-foot-in-front). For each separate animation I want to have a separate BMP file (using the same palette as all the other sprites) with the individual frames vertically aligned.

1 These values are merely an example, as are a lot of other ‘hard’ values. Only with proper play-testing and tweaking can we attain a balance in variables.