# Technical Design Template

**The Adventure of Captain Heart and Friends**

**Healthinauts**

### Hardware

1. Physical
   1. Standard graphics card
   2. Standard sound cards
   3. Input devices: keyboard and mouse
2. Drivers
   1. No special drivers needed

### Hardware abstraction layer

* 1. HAL
     1. Hardware abstracted by Flash

### User interface

1. Rather simple
2. Monitors Input status and stores it in the game data
3. Displays menus
4. Controls – keyboard and mouse
5. Should be reusable!!!

### Graphics engine

1. Simple level interface, tuned to a particular graphics and game type
   1. sprite based
   2. isometric

(more about this below)

1. Handles more complicated display aspects
   1. Overlays

### Sound and Music engine

1. Function of sound
   1. Effects to enhance reality
   2. Ambience
   3. Menu and Button sounds
   4. Clues about what is about to happen
   5. Narrating story of characters
   6. Right and wrong answer sound
   7. Heart Sounds
   8. Object sounds
   9. Character sounds
   10. Cheer sound of complete mini game
2. Kind of music
   1. Midi will be recorded into music tracks for music tracks
   2. Wave (high quality, lots of memory, fast) – all sound and music recording will be bounced to wave files to Flash
3. Simultaneous sounds
   1. Buffer management
   2. Streaming sound
4. Special features
   1. Headset (Binaural)
   2. Stereo (Transaural Stereo)
5. How to create
   1. Going to the recording studio
   2. Working out of home recording studio
   3. Live recording environment
   4. Using MIDI products –Reason
   5. Using real musicians

### Configuration system

1. Infinite retries
2. Skip option may be available

### Game Data

"A game is a database with a fancy interface"

1. Resources
   1. graphics models (sprites, characters)
   2. sounds, music
   3. images, backgrounds, video
   4. text
   5. Voice over
2. Level description
3. Game status
4. Event queue
5. High score menu
6. The game components communicate with the support components through the game data

### Event handler

1. Most games use an event based model
   1. Events take place
   2. Logic and Physics engine change the game status based on this
   3. New game status is displayed on the screen
2. Events
   1. User input
   2. Collisions
   3. Timers (controlled by the logic)
   4. Created by game tokens
3. Game tokens/object
   1. All entities in the game are tokens
   2. Tokens have a state
   3. These tokens react to events and create events
   4. Use state diagrams (finite-state machines) and interaction matrices to describe the behavior.
   5. (Don’t mistake with notion of object in object-oriented programming.)

### Logic engine

1. Handles all the game play
2. Enforces the rules
3. Contains the game AI

### Physics engine

1. Handles the simulation of the world
   1. collisions
   2. terrain changes
   3. pulses in the blood
   4. gravity
2. Limited or non-existent in simple games
3. Separation with logic engine is not always clear

### Program structure

This is the overall structure of the game and mini games.

1. Initialization
   1. Loading
   2. Intro
   3. Configuration
   4. Settings
2. Game loop
   1. Handle input
   2. Handle game actions
   3. Display
3. Finalization
   1. Saving score

### Timing

1. Target frame rate is 30 fps
   1. Flash handles double buffer
2. Rhythm mini game follows a timeline.