BooH – pre-production
Game Design Document

Updated: 2015-05-17, v1.0 (Final)

Contents

1. Game definition – mission statement

Booh is an arcade game where the player eats collectibles and avoids ghosts throughout mazes to survive. The game world is in space and the levels are floating islands. Power ups will give the player the ability to 'eat' enemies. The player can play both from a top view of the maze or from a first person view of BooH. In first person view the tension increases and the player doesn't know where the ghosts are.

2. Core gameplay

a. Main game view

The game is initially viewed from top-down and mainly shows a 3D view of a maze. Which is the active level. During playing the player can switch to First Person view mode and back, at any wanted time.

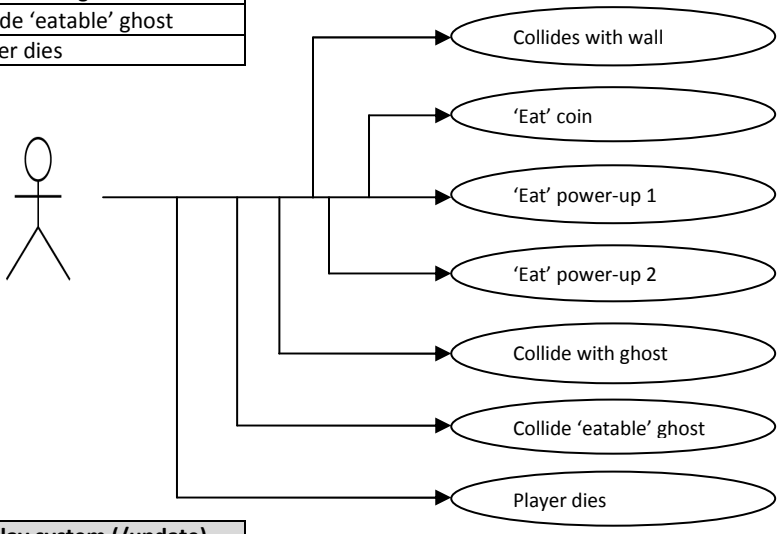
b. Core player activity

The player's task is to move through the maze and avoid getting eaten (= touched/ collides with the enemy ghosts). A level is finished when all collectibles are eaten. A coin is 'eaten'/ no longer shown in the maze as the user collides/ touches the coin. To illustrate: throughout finishing a level, there's are less and less collectibles visible in the maze.

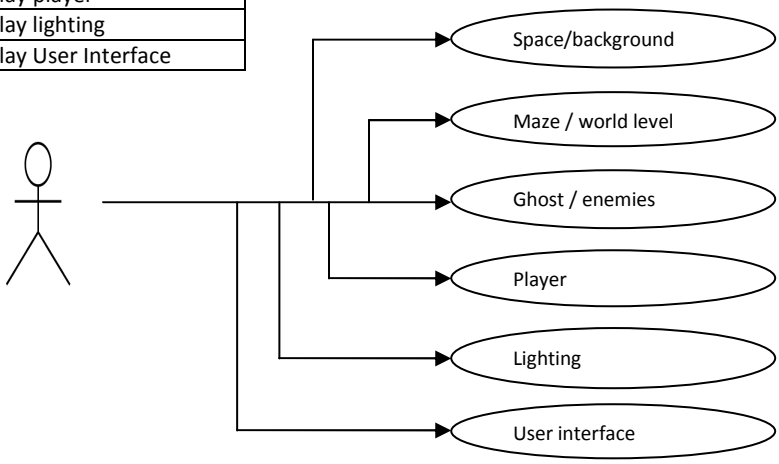
In top view the player automatically moves in a direction as the user presses an 'arrow key' on the keyboard and will keep moving automatically. The player moves till it bumps into a wall of the maze. To change the direction the player wants to move the player, the player simply pressed the appropriate arrow key on the keyboard. There is no support for joystick/ controller based movement.

In First Person view the player uses typical WSAD keys for moving around, combined with free mouse look. The player cannot move up and down (along the Y axis).

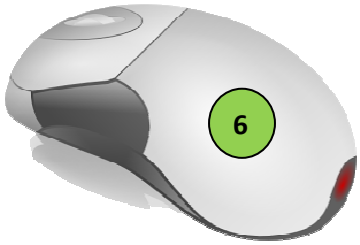
Game object interaction
Collides with wall
‘Eat’ coin
‘Eat’ power-up 1
‘Eat’ power-up 2
Collide with ghost
Collide ‘eatable’ ghost
Player dies



Display system (/update)
Display space/ background
Display maze/ world level
Display ghosts/enemies
Display player
Display lighting
Display User Interface



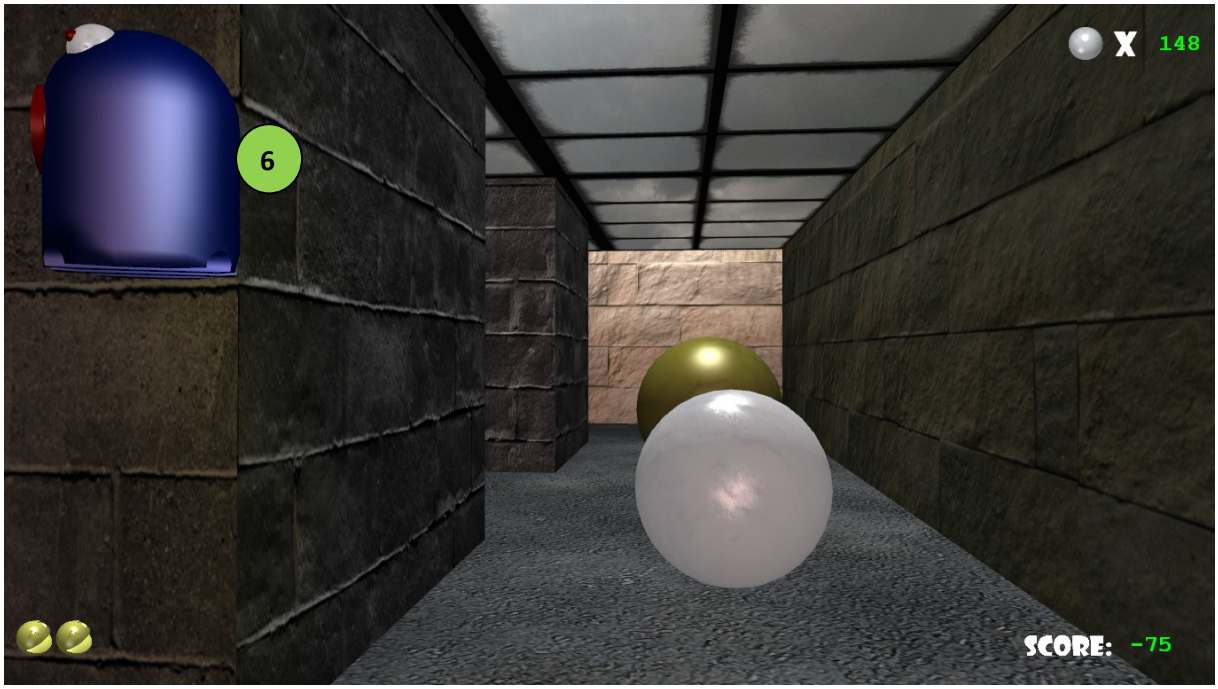
c. Controls



ID	Key(s)	Action	Game state
1	Arrow left/right/up/down	Move player	In-game, top view
1	Arrow left/right/up/down	Select item	Paused, menu
2	Enter	Confirm selection	Paused, menu
3	W, S, A, D	Move player	In-game, FP view
4	Space	Switch view mode	In-game, all views
5	Escape (ESC)	Go to menu	In-game, all views
6	Mouse movement	Free look around	In-game, FP view

d. In-game user interface: TOP view

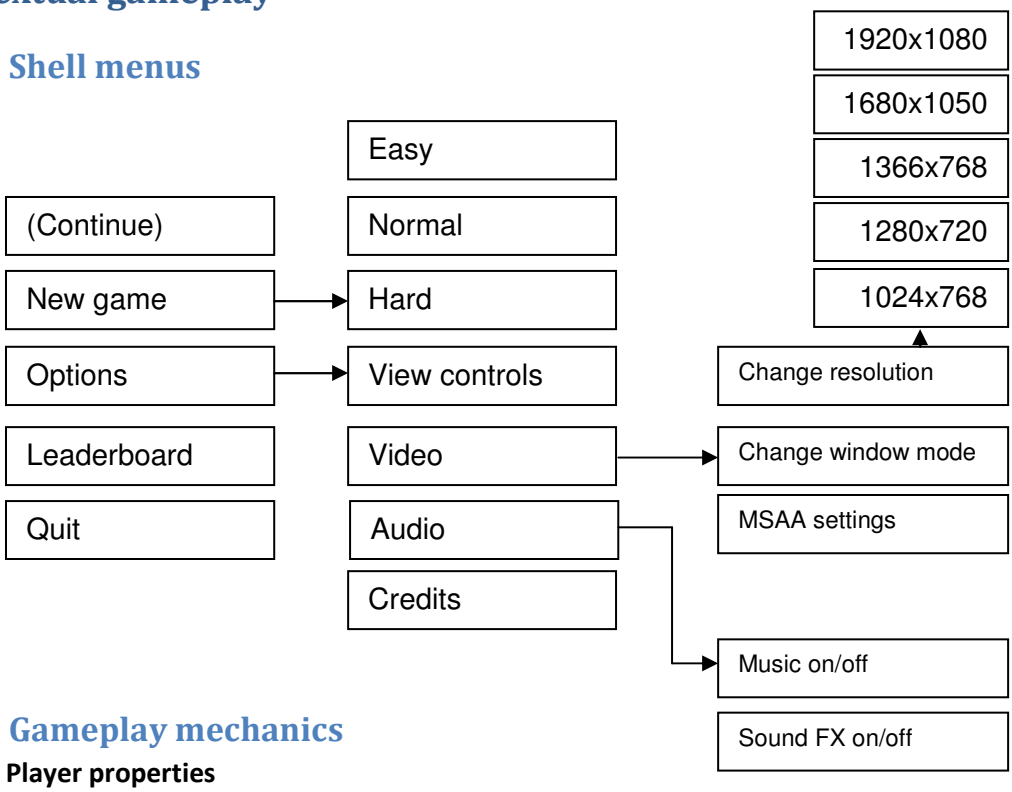
e. In-game user interface: First Person view



1. Collectibles remaining (in level)
2. Ghost indicator (rotate speed based on ghost nearby) => only in First Person
3. Remaining lives
4. Current score
5. The player
6. Ghost indicator (ghosts nearby or not) – FP view

3. Contextual gameplay

a. Shell menus



b. Gameplay mechanics

Player properties

- Has 4/3/3 lives at initial startup (easy/normal/hard)
- Dying costs 100/300/600 points (easy/normal/hard)
- Can move left/right and up/down through the maze (X and Z axis)
- Cannot go through walls and/or jump
- Floats 0.5m from the floor, top view (from origin of Player model)
- Floats 0.4m from the floor, FP view (from origin of Player model)
- Loses a live when colliding with a non-vulnerable ghost
- Eliminates a ghost when colliding with a vulnerable ghost
- Has a static starting position in each level
- Respawns at the initial starting position
- Extra life after level complete: *Normal* = **3**, *Hard* = **2** and **4**

Ghost properties

- There are 4 or 5 (active) ghosts in a level (blue, red, pink and orange)
- Blue ghost covers top left quarter of the maze
- Red ghost covers top right quarter of the maze
- Pink ghost covers bottom left quarter of the maze
- Orange ghosts covers bottom right quarter of the maze
- When a ghost dies it respawns after **5/6/7** seconds (easy/normal/hard)
- Ghosts have a static starting position in each level
- Will be vulnerable for ‘x’ seconds after player picks up BIG collectible

Ghost indicator

- Top view
    - Will light up when ghosts are vulnerable
- FP view
    - Will light up when ghosts are vulnerable
    - Will rotate fast when a ghost is nearby
      (the closer the ghost, the faster the rotation)

Sound

- Music
    - During menu state, looping
    - While viewing the leaderboard, looping
    - During normal gameplay in Top view mode, looping
    - During normal gameplay in First person view mode, looping
- Sound effects
    - Single sound when 'eating' a collectible
    - Single sound when 'eating' a power-up collectible
    - Single sound when eliminating a ghost
    - Single sound when selecting a menu item
    - Single sound when confirming a menu item
    - Single sound when starting a new game
    - Single sound when player has no more lives (and dies)
    - Single sound when level is finished

Difficulty

- Movement speed
    - Easy/normal/hard: player moves with speed: 0.037m per update (1/60 of 2.2m, locked)
    - Ghost speed factor: easy/ normal/ hard = player * 0.6/0.85/1.15

- Easy
    - Eating a regular collectible in top view gives **10** points
    - Eating a regular collectible in FP view gives **15** points
    - Eating a big collectible in top view gives **20** points
    - Eating a big collectible in top view gives **30** points
    - Killing a ghost in top view gives **100** points
    - Killing a ghost in first person view gives **1000** points
    - Ghosts vulnerable after power-up for **9** seconds

- Normal
    - Eating a regular collectible in top view gives **20** points
    - Eating a regular collectible in FP view gives **30** points
    - Eating a big collectible in top view gives **40** points
    - Eating a big collectible in top view gives **60** points
    - Killing a ghost in top view gives **200** points
    - Killing a ghost in first person view gives **2000** points
    - Ghosts vulnerable after power-up for **8** seconds

- Hard
    - Eating a regular collectible in top view gives **40** points
    - Eating a regular collectible in FP view gives **60** points
    - Eating a big collectible in top view gives **50** points
    - Eating a big collectible in top view gives **1000** points
    - Killing a ghost in top view gives **300** points
    - Killing a ghost in first person view gives **3000** points
    - Ghosts vulnerable after power-up for **6** seconds

Ghost AI

- Each ghost covers it's own quarter of the maze
  (levels with 5 ghosts: 5th ghost is not '*quarter-bound*')
- Each quarter has 4 or more possible target points
- A ghost is always moving the shortest way to it's current target
- When a target is reached, a next target point is chosen
- if player in ghost quarter + ghost vulnerable: target farthest from player
  OR if ghost not vulnerable: target closest to player
- If the player is not in the ghost's quarter, the ghost moves static from target 0 to 4. This is when ghosts are non-vulnerable
    - when ghostst become vulnerable the static route is revered (4 - 0)

c. Character background & game mood

Booh is hungry guy who loves eating collectibles.

The ghosts want to keep all collectibles for themselves and will do everything to prevent player from getting them. They will be terrified when BooH eats a power up (big collectible).

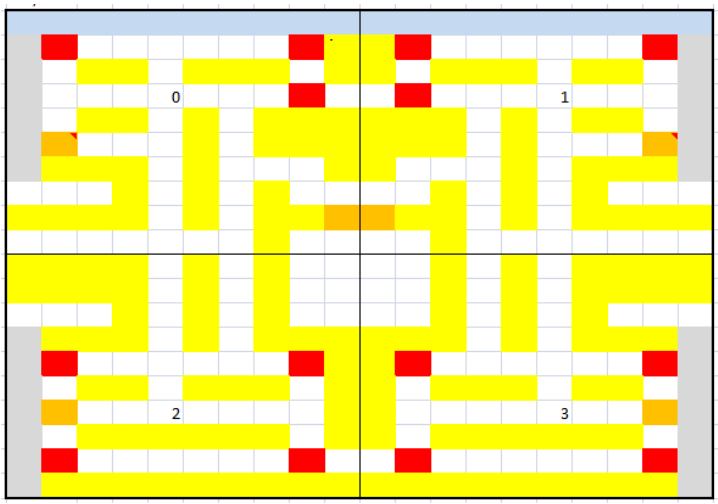
When playing in First Person mode, the moods get darker. Lights decrease and the tension gets higher with scary music. Where are the ghosts???

d. Level design

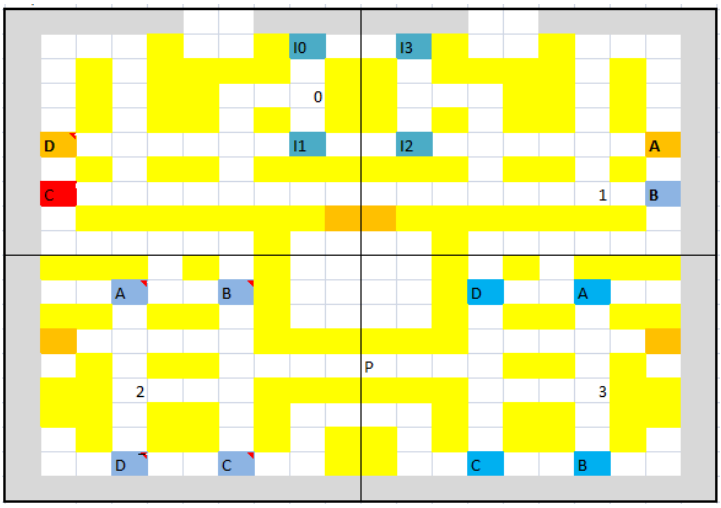
The initial BooH game will have 4 levels (might be more with expansions later).

Each level will have a unique number of collectibles and will always have 4 power-ups/ big collectibles. In each level both BooH and the ghosts have a static (initial) starting position. The music and sound will be the same throughout all levels.

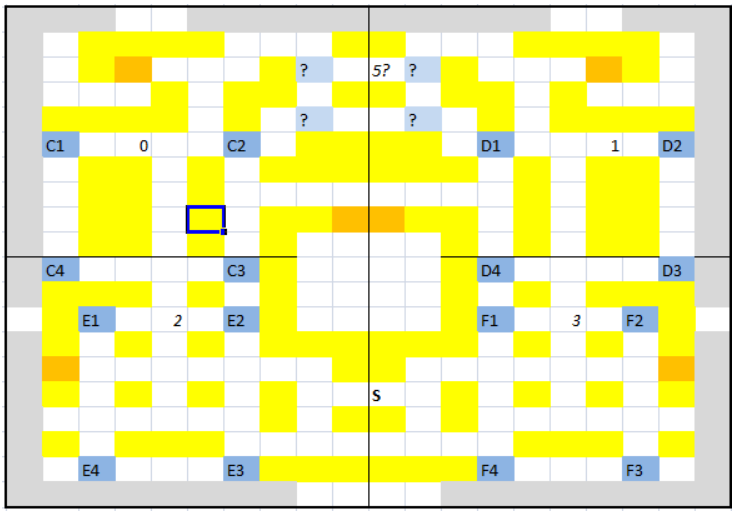
Level 1:



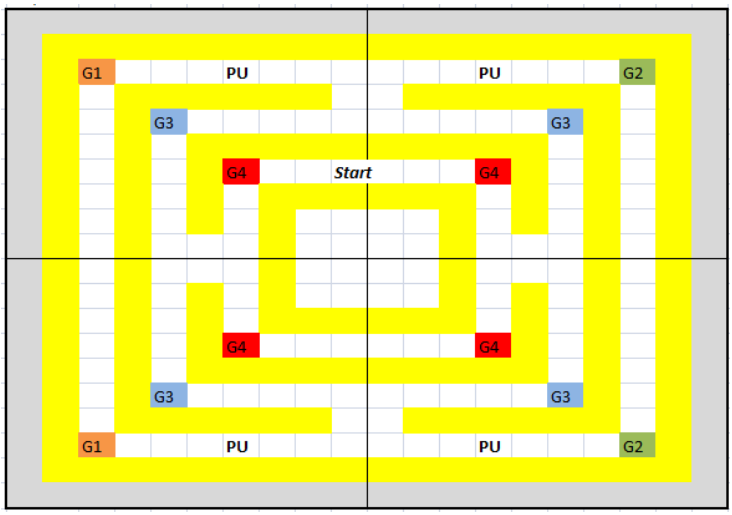
Level 2:



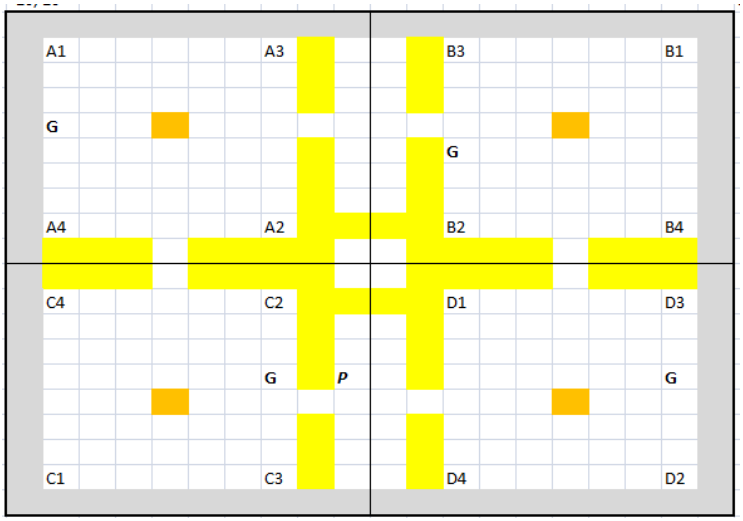
Level 3:



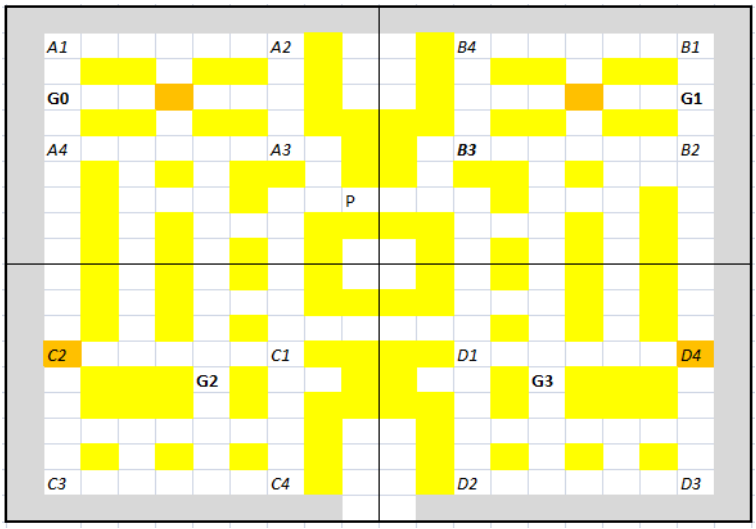
Level 4:



Level 5:



Level 6:



Reference:

e. Assets

Type	Description	Purpose	Filename
Sprite	Screenshot FP view	Menu	booh_fp.png
Sprite	Screenshot Top view	Menu	booh_top.png
Sprite	Ghost	Menu	ghost.png
Sprite	Arrow keys layout	Menu - controls	inp_arrowkeys.png
Sprite	Mouse controls	Menu - controls	inp_mouse.png
Sprite	Space key	Menu - controls	inp_space.png
Sprite	WSAD movement keys	Menu - controls	inp_wsad.png
Sprite	Main menu - game logo	Menu	menu.png
Sprite	Remaining collectible indicator	GUI	plr_coll.png
Sprite	Difficulty selection - collectible	Menu - new game	plr_collect.png
Sprite	Lives indicator in-game	GUI	plr_lives.png
Sprite	Score indicator in-game	GUI	score.png
Mesh	Player	Gameplay	player.x
Mesh	Collectible	Gameplay	collect.x
Mesh	Collectible PU (power up)	Gameplay	collectPU.x
Mesh	Ghost blue	Gameplay + indicator	ghost_B.x
Mesh	Ghost orange	Gameplay	ghost_O.x
Mesh	Ghost pink	Gameplay	ghost_P.x
Mesh	Ghost red	Gameplay	ghost_R.g
Mesh	Skybox	Gameplay	skybox.x
Mesh	Level **x** maze (base walls)	Gameplay	level**x**_base.x

Type	Description	Purpose	Filename
Mesh	Level **x** floor	Gameplay	level**x**_floor.x
Mesh	Level **x** roof	Gameplay	level**x**_roof.x
Mesh	Level switch base	Level done animation	lvl_sw.x
Mesh	Level switch 3D text 'level'	Level done animation	text_level.x
Mesh	Level switch 3D text 'next'	Level done animation	text_next.x
Mesh	Game over base scene	Game over animation	go.x
Mesh	Game over 3D text 'game'	Game over animation	t_game.x
Mesh	Game over 3D text 'over'	Game over animation	t_over.x
Texture	Ghost mouth	Mesh drawing	metal03b.dds
Texture	Ghost body red	Mesh drawing	glasssmooth_red.dds
Texture	Ghost body blue	Mesh drawing	glasssmooth_blue.dds
Texture	Ghost body pink	Mesh drawing	glasssmooth_pink.dds
Texture	Ghost body orange	Mesh drawing	glassmooth_orange.dds
Texture	Player body	Mesh drawing	glasssmooth_yellow.dds
Texture	Maze floor	Mesh drawing	aphalt2.dds
Texture	Maze roof	Mesh drawing	window_wh.dds
Texture	Maze 'ghost' step	Mesh drawing	rock.dds
Texture	Maze walls	Mesh drawing	grey_lime.dds
Texture	Maze wall variant 2	Mesh drawing	conc01b.dds
Cube Texture	Skybox, cube texture	Skybox mesh drawing	graycloud.dds
Texture	Collectibles (marble) + ghost eyes	Mesh drawing	ghost_marble.dds
Texture	Collectible PU	Mesh drawing	ghost_marble_yellow.dds
Texture	Concrete tiles (level done anim)	Mesh drawing	conctile.dds
Normalmap	Concrete tiles (level done anim)	Mesh drawing	conctile_NORM.dds
Normalmap	Maze wall variant 2	Mesh drawing	concr01b_NORM.dds
Normalmap	Collectibles (marble) + ghost eyes	Mesh drawing	ghost_marble_NORM.dds
Normalmap	Maze walls	Mesh drawing	grey_lime_NORM.dds
Scene file	Level 1 scene	Scene rendering	level1.scn
Scene file	Level 2 scene	Scene rendering	level2.scn
Scene file	Level 3 scene	Scene rendering	level3.scn
Scene file	Level 4 scene	Scene rendering	level4.scn
Scene file	Level done scene	Level done animation	switchscn.scn
Scene file	Game over scene	Game over animation	gameover.scn
Music	Menu music (loop)	Menu state	menu_loop.ogg
Music	Leaderboard (loop)	Viewing leaderboard	leaderboard_loop.ogg
Music	Gameplay top view (loop)	During top view gameplay	top_loop.ogg
Music	Gameplay FP view (loop)	During FP gameplay	fpmode_loop.ogg
Music	Gameover (no loop)	During G.O. animation	gameover.ogg
Sound	Eat collectible	Gameplay	chomp.ogg
Sound	Eat power-up collectible	Gameplay	chomp-pu.ogg
Sound	Eliminating ghost	Gameplay	eatghost.ogg
Sound	Confirm menu item	In menu state	menu.ogg
Sound	New game/ level start	When starting new level	startlevel.ogg
Sound	Game over	When lives = 0 and player dies	gameover.ogg
Sound	Level finished	When level is completed	leveldone.ogg

4. Technical Design documentation

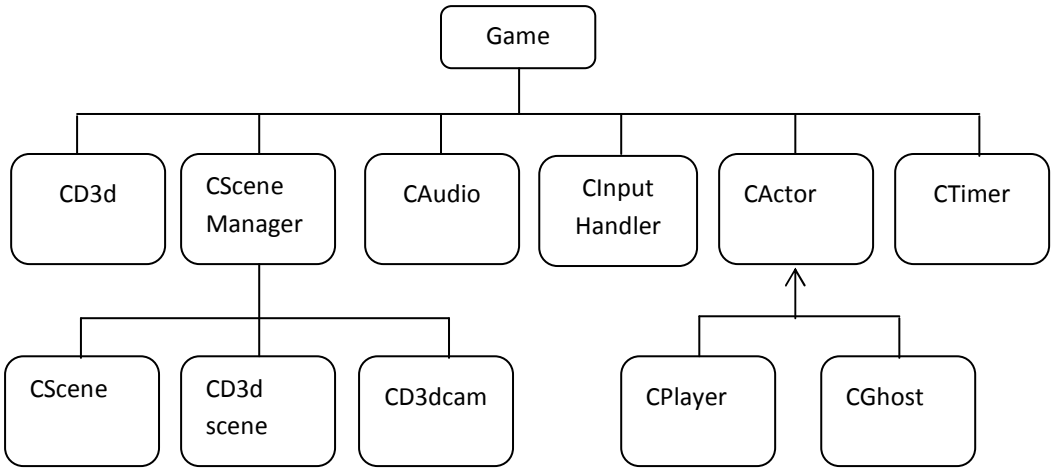
For convenience & the fact that there will be 1 developer, the Technical Design Documentation is added as a part of the Game Design Document (no separate document).

a. Main assumptions

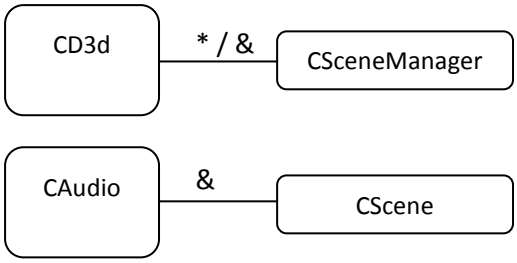
- The game uses the 'Crealysm' engine for rendering and input
- DirectX/Direct3D (9) is the used rendering API (shaders, no FFP)
- Audio is done using the FMOD API
- Input is done through the Crealysm input handler, using:
  keyboard -> Windows messaging, GetKeyStates
  mouse -> RAW input
- System requirements:
  Windows Vista and higher
  DX9c compatible graphics card, 256MB video memory
  512MB system memory
  250MB hard disk space

b. Class diagram(s) & dependencies

Excludes classes and relations that are not directly game-dependent
(IO, rendering; lighting, shaders, materials, fonts etc.)



Dependencies



c. Helper libraries

Namespace	Purpose
Crealysm_dxhelper	texture loading
Crealysm_dxmath	vectors, rays, bounding volumes
Crealysm_math	vectors
Crealysm_dxcollision	collision detection
Crealysm_general	error messages, text etc.

d. Implementation

- Level storage and processing
    - A levels is defined as a Crealysm scene, in the Crealysm file format
    - Collectibles are individual objects 'in the scene' (sharing a mesh)
    - Power-up collectibles share the collectibles mesh, scaled
    - Ghosts and player are individual objects 'in the scene'
    - The maze (floor + walls + ground) will be 3 individual meshes **Note;** where the walls mesh has sub meshes to be able to do AABB/Sphere collision detections
    - The ghost indicator will use one of the existing ghost meshes
    - The mesh instance ID's of the ghosts, player, ghost indicator and collectibles will be static (for simplicity):

Inst	Object	Point light ID
0	Maze base (walls)	
1	Maze floror/ground	
2	Maze roof (FP only)	
3	Player	6
4	Ghost indicator	
5 (X)	Ghosts	7 - onwards
X - X+4 (Y)	Power-up collectibles	1 - 4
Y – end	Collectibles	

- Player and ghost properties are stored in their inherited player and ghost classes. This is done hardcoded (not data driven)
- Ghost AI is done hard coded
- Switching levels is done using the Crealysm scene manager
- Game state is done through a customized version of the Crealysm game state manager. States: *menu1/2/3/4, in-game, leaderboard and controls view*
- Handling input
    - Keyboard and mouse input is captured in main Windows messaging (WINPROC) and stored in CInputHandler class
    - CGame class retrieves input from CInputHandler class
- Handling game logics is done in game specific member functions within the CGame class

e. Coding standards

- All class names start with 'C' followed by one capital for the name
- Strict split in public and private members, private members 'set/get'
    - With the exception of the **CPlayer** and **CGhost** classes
- Classes who 'need' each other will be done through forward declaration using pointers or const reference (if possible)
- Class interface
    - First variables, then functions
    - First const, then non-const
- Class (and other) function implementations always have a comment header, telling: goal of the function, description what the function returns
- A class should serve only 1 purpose
- All game input processing and updating is all done within the CGame class (this includes the game's global variables)
- Member variables are recognizable by starting with '**m**X…'
- Global function names and local variable names are *lower camelCase*
- Global variables names are fully lowercase
- Defines are fully UPPERCASE
- Member function names are *upper CamelCase*
- There will be no memory leaks (checked by *Visual Leak Detector*)
- Direct3D debugging is used and set to '*stop on D3D9 error*'

f. Tools

- Engineering: Visual Studio 2010 Ultimate
- Modeling: Autodesk 3D Studio Max 2011 (student)
- Scene and IO editing: notepad
- Textures and more: Paint.NET
- Mapping: Shadermap2

5. Game requirements

Requirement	Priority (must/like/nice)	Category (process/gameplay/production)
Player movement	Must have	Gameplay
Ghost AI	Must have	Gameplay
Collision detection top view	Must have	Gameplay
Collision detection FP view	Must have	Gameplay
Gathering collectibles	Must have	Gameplay
Power-up/ make ghosts vulnerable	Must have	Gameplay
Level loading	Must have	Production
Level switching	Must have	Production/ gameplay
Game/ menu states	Must have	Production
Switching top/FP- view (and back)	Must have	Production
Ghost indicator top view	Like to have	Gameplay
Ghost indicator FP view	Like to have	Gameplay
Scoring system	Like to have	Gameplay
Replay function	Nice to have	Gameplay
Wall sliding in FP-view	Nice to have	Gameplay
Difficulty levels	Like to have	Gameplay/ production
Leaderboard	Like to have	Production
Dynamic skybox	Like to have	Production
Dynamic audio (volumes)	Like to have	Production
Animated (camera) switching view	Nice to have	Production
Audio menu, options (i.e. volume)	Nice to have	Production
Level editor	Nice to have	Process/ production

6. Major milestones

	First playable	Alpha	Code freeze	Beta	Code release candidate
Timeframe	+1 month	+2 months	+2,5 months	+3 months	+4 months
Art	All models simplified/concept	Player + ghosts ready. Rest concept	Player + ghosts ready. Levels ready. Rest concept	All done	All done
Engineering	Basic functionality ready: playable, no scoring/menu's/AI	Playable with AI. Scoring ready. Menu's/ AI not ready	Feature complete, except menu's/ states	Only bug fixing	Only crash bug fixing
Design	Game design document ready	Game design document ready	Feature list final	n/a	n/a
Sound	All placeholders	FX ready. Music placeholders	FX ready. Music placeholders	FX + Music ready	FX + Music ready
Production	Level loading. Switching view mode	Level loading. Switching view mode	Level switching added	Only bug fixing	Only crash bug fixing
QA	n/a	Start play testing	Play testing finished. Final QA plan ready	Start QA execution. Start Beta testing	All done

7. RISK analysis

Risk	Likeliness	Measure
Crealysm engine not fully ready for needed features	Medium	Expand engine functionality 'on the fly' or solve hard coded for game
No member for final modeling	High	Find one before Alpha ☺
No member for music	High	Find one before Alpha ☺
Not enough experience on AI engineering	High	Gain needed knowledge / re-use 3rd party / find additional team member

Notes on Milestones, updated 15-4-2015:

- Started development 2nd week of December
- Finished 1st playable 17th of January
- Aiming for Alpha release 2nd week of February (+1,5 weeks)
- Alpha deadline not met, moved to 1st of March (+ 2 weeks)
- Alpha delivered 1st of March 2015

- Beta delivered: 19th of April 2015 (+ 1 month total)
- Beta build 2: 27th of April 2015

- Final release 1.0 delivered 17th of May 2015