

Metal man

Group: Not Home Alone

Rittichai Sriubolmas 57070503429

Thitiya Trithipkaiwanpon 57070503449

Saranyu Pienpitak 57070503462

Tissatat Charoenchai 57070503469

**Introduction**

Purpose of design document

“Metal-man” is a PC game application. Our main purpose on this project is to create an entertaining program for people. The game presents mix gameplay from many good old games such as wall gliding, dashing which we get an idea from “mega man” series and side scrolling mechanic from “metal slug”. In the future developer we aim to put in more mechanics that will improve gaming experiences also we will add more environment for player to play. In this game user have an ability whether they want to login or not but we suggest that they should log in for compete which another player on leader board which will be update everyday also player can share their progress on Facebook if they want.

Access – After start game user must enter the character name before play this game

Access Validation – No, because it is offline game.

Document conventions

|  |  |
| --- | --- |
| Acronym | Definition |
| Inventory | It is a place that keep any items that player find and get them |
| Skill | Ability that user can use to fight with the enemy |
| EXP | Experience that user acquire from monster by killing it that make your character more strength |
| HP | HP : Health point, if it drops down to 0 your character will die |
| Potion | Item that use to heal your HP |

Scope

The scope of the project is consists of a fully functional main menu, action performing, result, report, and character movement which each part will have their own uniqueness. Main menu, a user interface that give an access ability to the user to access another part of the program. Action performing, ability for user to use a combination of button to make one big action. Result, to show the overall or in-depth result after player beat the game or failed. Report, will grant user permission to send back any bug or problem during their game play experience. Movement, focus on character movement mechanic.

Document overview

Introduction: It is a part that explain the necessary thing of the document which is the background of software and purpose of software.

The data design: Describes structures that reside within the software. Attributes and relationships between data objects dictate the choice of data structures.

The architecture design: Uses information flowing characteristics, and maps them into the program structure. The transformation mapping method is applied to exhibit distinct boundaries between incoming and outgoing data. The data flow diagrams allocate control input, processing and output along three separate modules.

Component-level design: Describes any component that occur in program and interface between component and user interface

The interface design: Describes internal and external program interfaces, as well as the design of human interface. Internal and external interface designs are based on the information obtained from the analysis model.

Process Manual Specifications: Describe about project planning of our software to know how it going on.

**Data/Class design**

Internal software data structure

**Administrators**

* Can add and update new data in the software by sending data to the database and update continuously.
* Can access in each information of this system correctly.

**Users**

* + - When user keep or collect any item in game. Then each item that is kept will be added into the database of the user.
    - User can select option to manage the video, sound and control. So the system will query any detail in each option from database for user to select it.
    - User can view full map.
    - User can use skill in the game to fight with the monster.
    - User be able to use item from the inventory of user.

Global data structure

The global data structure retrieves data from the Metal man database, which is a data that every components can access and inspect value and used as data sharing. When there are any changes it will be recorded to the database. Such as detail of monster.

Temporary data structure

Browser collect information of each user that download this game the called “cookie” by keep in the user computer.

Database description

Database system that design for making this game.

|  |  |
| --- | --- |
| Table Name | Item |
| Attribute | * Item name * Description |
| Description | A table that use to collect any item data in the game |

|  |  |
| --- | --- |
| Table Name | Character |
| Attribute | * Character name * Level * Attack * HP * EXP |
| Description | A table that use to keep every information of character in the game. |

|  |  |
| --- | --- |
| Table Name | Map |
| Attribute | * Map name * Monster name |
| Description | A table that use to show the map that has which monster. |

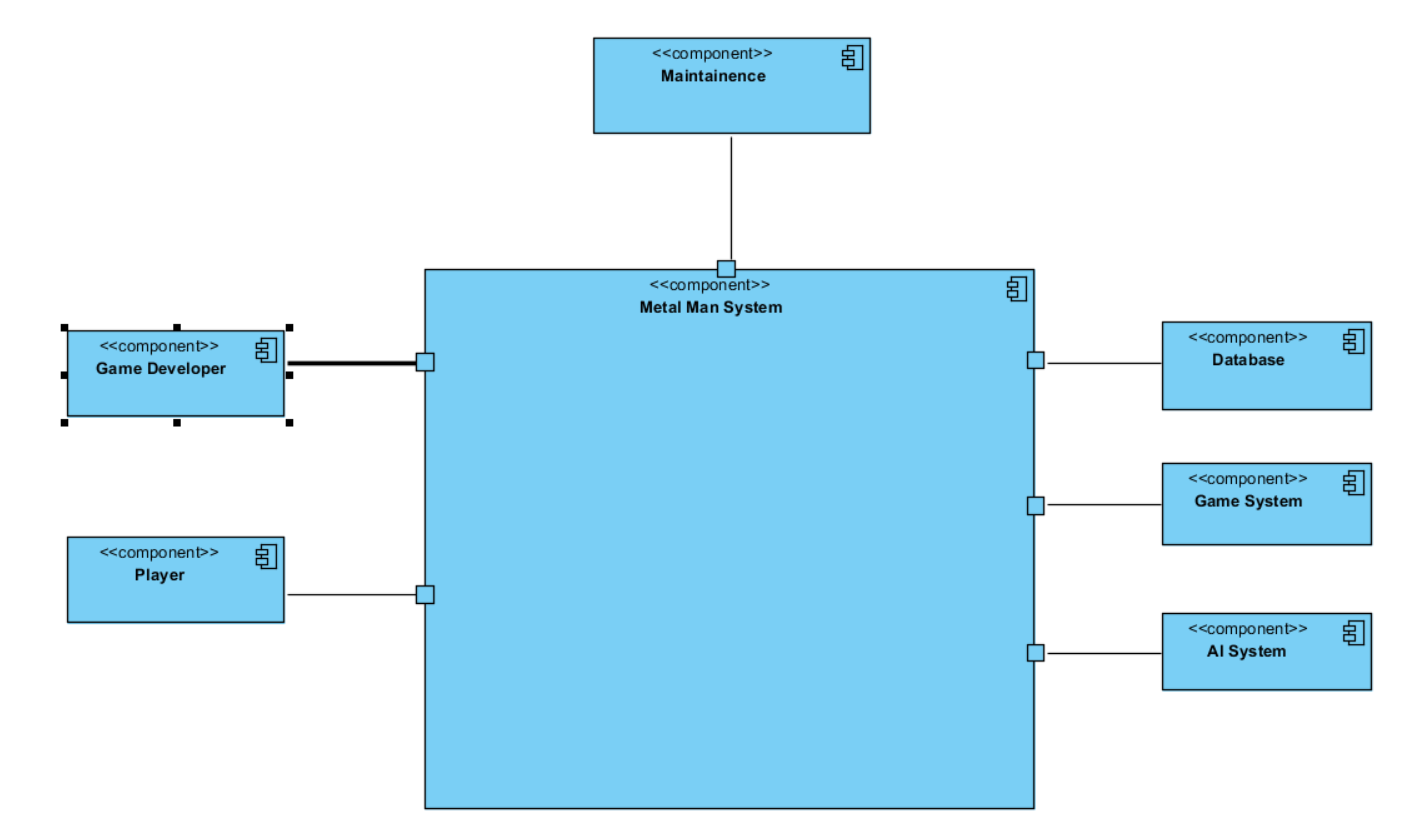
|  |  |
| --- | --- |
| Table Name | Inventory |
| Attribute | * Item name * Quality * Description |
| Description | A table that use to keep data of item for each user collect in game. |

|  |  |
| --- | --- |
| Table Name | Option |
| Attribute | * Video * Sound * Control |
| Description | A table that use to setting sound, video graphic and controller in the game |

|  |  |
| --- | --- |
| Table Name | Skill |
| Attribute | * Skill name * Damage * Description |
| Description | A table that use to keep data of skill. |

**Architectural design**

**Architectural design (level 1)**

UML Architectural Context Diagram

Metal systems are interfaced with other systems by:

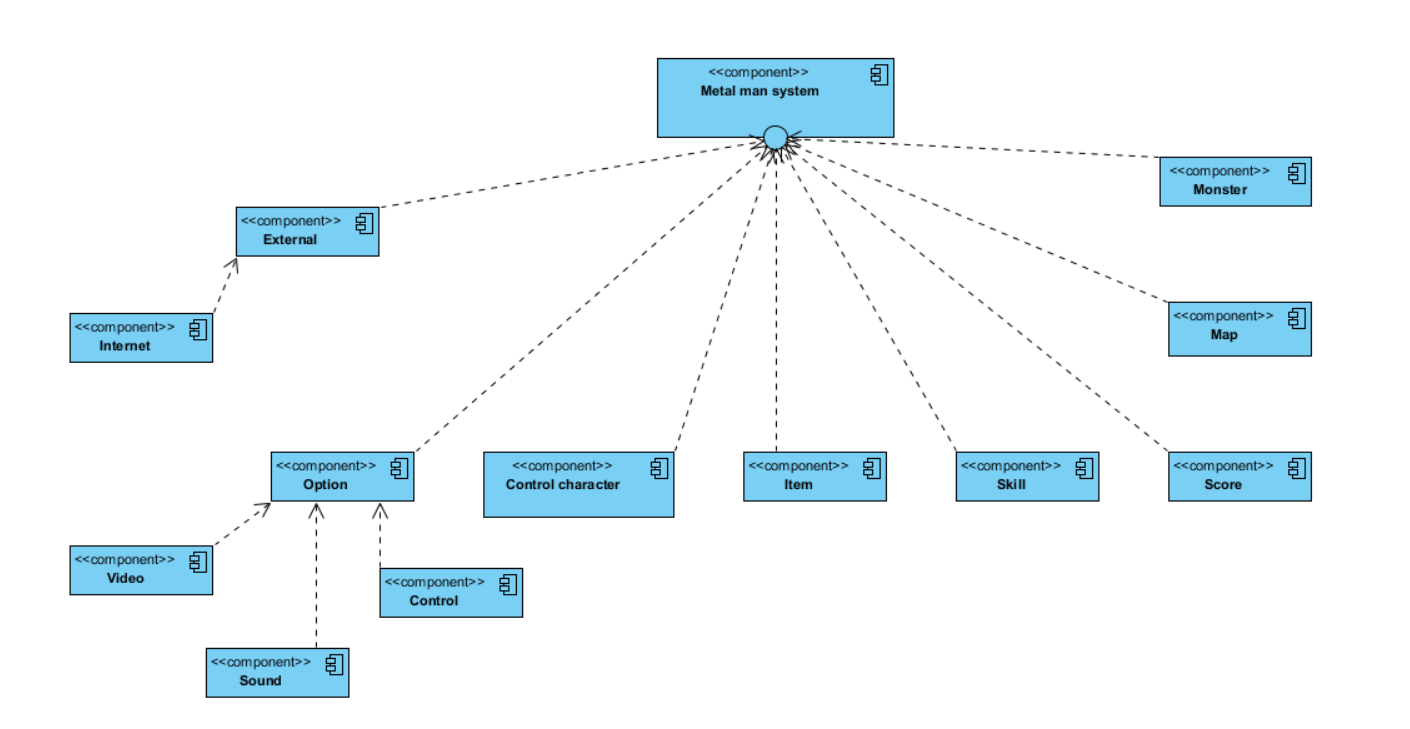
**Actor**

* Game Developer: A person that use this system to check bug and test the program and also add some new detail and manage information in the database.
* Player: use this system to run the game and play such as control movement of the character and attack the enemy.

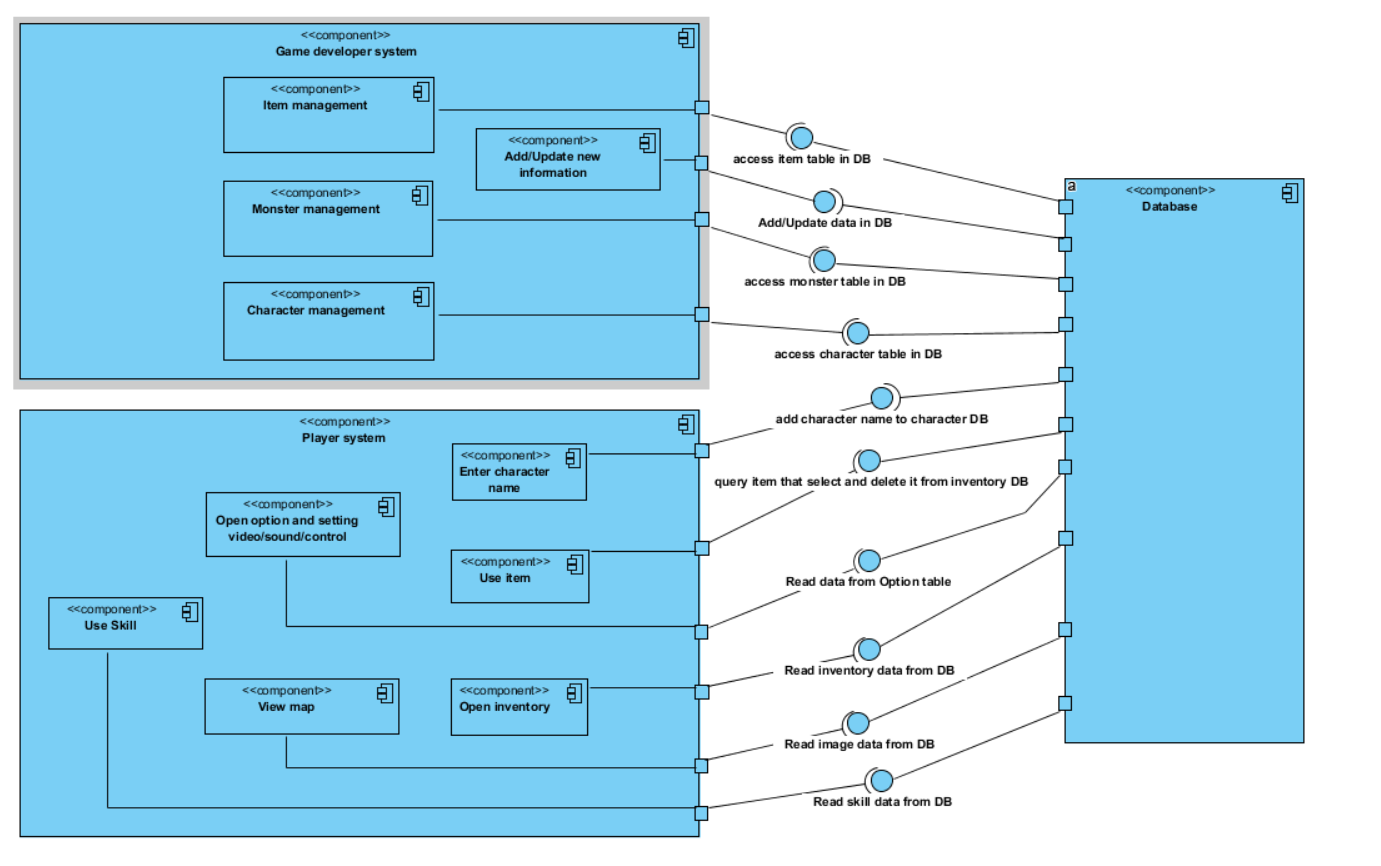
**Peers**

* Database: The database serves to maintain all games information and make a use of share information in the database system.
* Game system: System that cover overall the gameplay.
* AI system: System that make any monster walk and attack automatically.

UML Component level diagram

Component structure

Component diagram

Our system will separate to 2 main components such as **Game developer site** and **Player site**.

* Game developer site is a program that use for admin to work such as add/delete information, manage information and check some bug and fix. So it has sub component:
  + a. Item/Monster/Character management: It is a component that admin want to manage the main information in the game. So admin can edit some information that make the game is bad.
  + b. Add/Update new information: Using to add new information in the game such as new monster or new map.
* Player site is a program that use for user who want to play this game, they can use many option in this game such as fight with the enemy, use some item, collecting item, view map and using skill etc. So in this component will consist of:
  + Enter character name: It is component that make user to create character by enter character name to enter the game.
  + Open inventory: This one is a component that has a task to query every item in the inventory database and show it to the user view in game.
  + Use item: A component that use to query an item that user select in the game from inventory database and after use that item program will delete that item in the inventory database.
  + View map: A component that use make a person can see the map in the game by query image in map database.
  + Use skill: A component that is a special skill that deal damage to a monster that activated by user command.
  + Open option and setting video/sound/control: A component that allow user to set their video, sound and control in the game to make more comfortable for each user.

Component-level design:

1. Description component 1: Main menu
   1. Processing narrative (PSPEC) for component: main menu

This component is to give user an option to do anything. User have option to play the game, setting the program, view credit, or quit the program (properly)

* 1. Component 1 processing detail
     1. Design Class hierarchy for component: main menu

|  |
| --- |
| MAIN MENU |
| -playButton  -optionButton  -creditButton  -quitButton |
|
|
|
| +startGame()  +gotoOption()  viewCredit()  quitProgram() |
|
|
|

* + 1. Restrictions/limitations for component: main menu
       1. When user click (choose) result will happen suddenly and go to another page or quit the game.
       2. Since result occurs suddenly, user can not choose more than one option.
       3. There are some button which is special command in window, it might interfere work progress.
    2. Performance issues for component: main menu
       1. If program don’t run in proper environment, it could slow, have error, or even can’t run.
    3. Design constraints for component: main menu
       1. To choose an option, user must click or push the right button in order to choose it.
    4. Processing detail for each operation of component: main menu
       1. Processing narrative (PSPEC) for each operation

|  |  |
| --- | --- |
| Operation | Description |
| startGame() | Choose to start the game and go into the game. |
| gotoOption() | go to option menu to adjust game |
| viewCredit() | view the credit of game development team |
| quitProgram() | close the program |

* + - 1. Algorithmic model (e.g., PDL) for each operation

|  |
| --- |
| startGame()  if click position = play button location  run the game  end if |

|  |
| --- |
| gotoOption()  if click position = option button location  run the game  end if |

|  |
| --- |
| viewCredit()  if click position = credit button location  run the game  end if |

|  |
| --- |
| quitProgram()  if click position = quit button location  run the game  end if |

* 1. Component: main menu test points list and description

|  |  |  |
| --- | --- | --- |
| No. | test description | expected result |
| 1 | press outside the button | response nothing |
| 2 | press ‘play’ button in main menu | start the game |
| 3 | press ‘option’ button in main menu | go to program adjusting option page |
| 4 | adjust (side scrolling) in option | increasing or decreasing sound or light of program (according to adjusting tag) |
| 5 | press ‘credit’ button in main menu | show credit page, which contain list of team member and their task |
| 6 | press ‘quite’ button in main menu | execute the program and close it |

* 1. Component 1 dynamic behavior
  2. Component 1 interfaces

The user interfaces of the component will be shown on page

1. Description component 2: Option and game setting
   1. Processing narrative (PSPEC) for component: option and game setting

This component allow user to adjust the game and program. User can adjust music of main menu, music within game, sound effect, and light.

* 1. Component 1 processing detail
     1. Design Class hierarchy for component: option and game setting

|  |
| --- |
| OPTION AND GAME SETTING |
| -master volume  -music volume  -sound effect volume  -light |
|
|
|
| +adjustMast()  +adjustMusi()  +adjustSfx()  +adjustLight() |
|
|
|

* + 1. Restrictions/limitations for component: option and game setting
       1. Option adjustment are controlled by phone’s setting. For example, the game will still quiet if phone set as that.
    2. Performance issues for component: option and game setting
       1. The level of adjustment is only ratio of that setting, so player should prioritize on result not shown value.
    3. Design constraints for component: option and game setting
       1. All sound setting are under control of master volume.
    4. Processing detail for each operation of component: option and game setting
       1. Processing narrative (PSPEC) for each operation

|  |  |
| --- | --- |
| Operation | Description |
| adjustMast() | adjust program sound level |
| adjustMusi() | adjust program and game music |
| adjustSfx() | adjust sound effect of program and game |
| adjustLight() | adjust the light, program projecting |

* + - 1. Algorithmic model (e.g., PDL) for each operation

|  |
| --- |
| adjustMast()  value = getBarMast();  setMast(value/max); |

|  |
| --- |
| adjustMusi()  value = getBarMusi();  setMusi(value/max); |

|  |
| --- |
| adjustSfx()  value = getBarSfx();  setSfx(value/max); |

|  |
| --- |
| adjustLight()  value = getBarLight();  setLight(value/max); |

* 1. Component: option and game setting test points list and description

|  |  |  |
| --- | --- | --- |
| No. | test description | expected result |
| 1 | slide master volume bar | adjusting master volume, increase all sound if slide to right, decrease if slide left |
| 2 | slide master volume bar to leftmost | disable all sound |
| 3 | slide music volume bar | adjusting music volume,  increase music if slide to right, decrease if slide left |
| 4 | slide music volume bar to leftmost | disable music |
| 5 | slide Sfx volume bar | adjust sound effect volume,  increase sound effect if slide to right, decrease if slide left |
| 6 | slide Sfx volume bar to left most | disable sound effect |
| 7 | slide light volume bar | adjust light  Increasing light level of program, rightmost is maximum level. Decreasing light level of program, leftmost is minimum level |
| 9 | press ‘back’ button in any menu | go to previous page in most case main menu |

* 1. Component: option and game setting dynamic behavior
  2. Component: option and game setting interface(s)

The user interfaces of the component will be shown on page

1. Description component 3: In game controlling
   1. Processing narrative (PSPEC) for component: in game controlling

This component is the controlling of game. Since our game is shoot-and-slash 2D game, the control will be go left, go right, jump, crouch, shoot, and slash, to fight through enemy.

* 1. Component: in game controlling processing detail
     1. Design Class hierarchy for component: in game controlling

|  |
| --- |
| IN GAME CONTROLLING |
| -leftButton  -rightButton  -jumpButton  -crouchButton  -shootButton  -slashButton  -HP  -point  -hostileObject |
|
|
|
| +goLeft()  +goRight()  +jump()  +crouch()  +shoot()  +slash()  +hurt()  +pointing() |
|
|
|

* + 1. Restrictions/limitations for component: in game controlling
       1. The game can have bug, and the bug may affect controlling. The result are from interact to ‘special situation’ to can interact to game at all.
       2. player must control correctly to get wanted result(press the right button)
    2. Performance issues for component: in game controlling
       1. Some data (in this case button) can be lost if user press too many button at a time.
       2. If program don’t run in proper environment, it could slow, have error, or even can’t run.
    3. Design constraints for component: in game controlling
       1. player must understand basic element of game by himself(or reading manual) to know which one is playing-character, enemy, map, obstacle, hostile object, or how to pass the game
       2. the action speed of everything in this game are under game developer providing
       3. HUD are fixed, so some object can be concealed by HUD.
    4. Processing detail for each operation of component: in game controlling
       1. Processing narrative (PSPEC) for each operation

|  |  |
| --- | --- |
| Operation | Description |
| move() | move character to left or right with moving animation |
| jump() | move character to up then fall |
| crouch() | the character change standing gesture to crouching(lower level) |
| shoot() | fire a projectile that damage when hit enemy |
| slash() | damage enemy in front of character |
| hurt() | recuse health point that display by bar |
| pointing() | increasing the point by time passing, and enemy eliminated |

* + - 1. Algorithmic model (e.g., PDL) for each operation

|  |
| --- |
| move()  while press right || press left  animation(moving);  if new direction != old direction  walkReverse(direction);  end if  if front space > obstacle\_limit  walk(direction);  end if  end while |

|  |
| --- |
| jump()  if press jump  animation(jumping);  jumping();  move();  end if |

|  |
| --- |
| crouch()  while press crouch  animation(crouch);  crouch();  move();  end while |

|  |
| --- |
| shoot()  while press shoot  animation(charging);  chargeProjectile();  end while  releaseProjectile(); |

|  |
| --- |
| slash()  if press slashing  animation(slashing);  slashDamaging();  end if |

|  |
| --- |
| hurt()  if hostile colide character  animation(hurting);  getDamage();  HP = HP - damage;  end if |

|  |
| --- |
| pointing()  while not game over  point++  if enemy die  point = point + enemyPoint  end if  end while  return point; |

* 1. Component: in game controlling test points list and description

|  |  |  |
| --- | --- | --- |
| No. | test description | expected result |
| 1 | get hit by enemy or hostile object | the health bar decrease |
| 2 | player lost all health | game over, display score, take player to scoreboard, then asker player to play again or go to main menu |
| 3 | player press pause option | pause the game, ask player again for confirmation, go to main menu if yes, stay in game if no |
| 4 | character fall of the stage | instant death, do ‘game over’ process |
| 5 | character walk to obstacle | character cannot move farther in that direction until it jump over |

* 1. Component: in game controlling dynamic behavior
  2. Component: in game controlling interface(s)

The user interfaces of the component will be shown on page

1. Description component 3: Scoreboard managing
   1. Processing narrative (PSPEC) for component

In this component, when player finish the game (or game over), game will bring user to score screen. If user’s score is top 20th, game will as the name to be display on scoreboard (like old arcade game)

* 1. Component 1 processing detail
     1. Design Class hierarchy for component: scoreboard managing

|  |
| --- |
| SCOREBOARD MANAGING |
| -playerName[20]  -playerScore[20] |
|
|
|
| +save()  +clear() |
|
|
|

* + 1. Restrictions/limitations for component: scoreboard managing
       1. if saving fail, no score will be save
       2. all score be lost, if save file damaged, clear, or lost(automatically create new save)
       3. Naming must be between 1 to 10 character (and only contain letter, number, !, @, #,$, %, &, +, -, \*, /, (, ) )
    2. Performance issues for component: scoreboard managing
       1. if player does not successfully done, the game won’t record
    3. Design constraints for component: scoreboard managing
       1. score will only save and display top 20 score
    4. Processing detail for each operation of component: scoreboard managing
       1. Processing narrative (PSPEC) for each operation

|  |  |
| --- | --- |
| Operation | Description |
| save() | save the score with name asking from player |
| clear() | delete all score memory |

* + - 1. Algorithmic model (e.g., PDL) for each operation

|  |
| --- |
| save()  i = 0;  while not confirm  input char  if char available // letter, number, !, @, #,$, %, &, +, -, \*, /, (, )  name[i] = char  i++  end if  if pres reset  clear(name);  i = 0;  end if  end if  score = getScore();  rank = getRank(score);  updateRank(rank);  save(name,rank); |

|  |
| --- |
| clear()  if confirm  for rank=1; rank<= 20; rank++  clearScore(i);  clearRank(i)  end for  end if |

* 1. Component: scoreboard managing test points list and description

|  |  |  |
| --- | --- | --- |
| No. | test description | expected result |
| 1 | player’s score is not top 20 | display scoreboard, and ask player to play again or go to main menu |
| 2 | player’s score is top 20 | take user to their ranking location, then ask for saving name, then ask player to play again or go to main menu |
| 3 | user press clear button | ask to confirm decision, then all score and saved name will be cleared, user are still in the scoreboard but blank |
| 4 | user quit before successfully save the score | quit this window as they wanted, and don’t save anything |

* 1. Component: scoreboard managing dynamic behavior
  2. Component: scoreboard managing interface(s)

The user interfaces of the component will be shown on page

USER INTERFACE DESIGN

1. USER INTERFACE DESIGN RULES.
   1. Strive for consistency: we control the continuity of design, to create common theme from font concept and color. So the interface will express itself as a same set and format.
   2. Cater to universal usability: we try to satisfy every user, so we gathering information by interviewing some example. Then we choose the group that most massive and possible to accomplish.
   3. Offer information feedback: we always consider something to responds when user interact with program. It prevent user to feel alone and uninterested by keep stimulate attendance from user.
   4. Design dialog yield closure: we create story, included beginning, sequence, conclude, so that player got drawn attention and try to discover our story.
   5. Prevent error: user usage lead to various possible error. Se we as designer must prevent any error as much as possible. Fortunately, our program is not fragile and don’t interact with serious issue. However, we put some safety measurement to some important choices.
   6. Permit easy reversal of actions: our program is free and easy to reversal action. But there will limit of reversing to keep our main design theme.
   7. Support internal locus of control: we make an option to manage internal locus, and we will feed back the result of action. So user will know the consequence of action.
   8. Reduce short-term memory load: we reduce length of thing we present in order to optimize short-term memory.
2. COMPONENTS AND DEVELOPMENT TOOLS USED.
   1. Justinmind Prototyper: design prototype and user interface
   2. Unity: game engine, finished program used to create game
   3. Atom text editor: text editor tool to coding the game
   4. Visual Paradigm: the program for draw diagram and planning
3. SCREEN IMAGES AND DESCRIPTION. This section can be included in the Component-Level Design, if desired
   1. main menu, the first page user see



* 1. in case user choose option
     1. when user choose option, user will brought to option, to choose whether they want to adjust video, sound, or control



* + 1. if user choose to adjust video, program will take user to video adjusting page
    2. if user choose to adjust sound, program will take user to sound adjusting page. In this page, user can set master volume, music volume, or sound effect volume.
    3. if user choose to adjust controlling, program will take user to control setting page. This page will display all key binding, and user can change the key for comfort.
  1. in case user choose credit
     1. when user choose credit, program will show the credit page. The page include all staff and their task. When user done, just click back and return back to main menu.



* 1. in case user choose start
     1. The program will start the actual game. In game, there is health bar on the left-top-corner with score next to. The character will be in the middle.
     2. Scoreboard will show up when game over. This screen include, title, best score player which ask user name if they got ranked, clear score button, save button, and next which will link user to main menu.
  2. in case user choose exit, the program will terminate and close down.

6. OTHER INTERFACES DESIGN

1. Hardware interfaces - offline computer, use computer as processing, interfacing, and store all data. There is no internet needed, just save as file inside the computer.
2. Software interfaces design - we use Unity as the game engine for the game
3. Communication interfaces - there is only saved file to store the score of player

7. How to use each scenario

Scenario 1: Accept term condition

1. Player will access in to the main menu first

2. Player accept the term condition to play our game.

Scenario 2: Start playing

1. Player Click the start game to entering the game

2. Player enter his/her character name

3. Player begin to play

Scenario 3: Play the game

1. After game start player can control his/her character in this game.

2. Players play the game by using every button include in this game.

Scenario 4: Change option and setting game

1. Player open option menu to setting sound/video/control in game.

2. Player choose and change every detail that they want such as decrease sound.

3. Player click OK to change it.

Scenario 5: Use item and skill

1. Players open their inventory menu and select item that they want to use.

2. Click the skill in skill tab to activate skill

Scenario 6: Get the total score

1. Player finish the game or game over (character died).

2. Score will print out to the screen to the user.

3. Game is end.

Scenario 7: Quit the game

1. Player open menu in game.

2. Player select quit button to quit the game and get back to the main menu.

Scenario 8: Admin edit program.

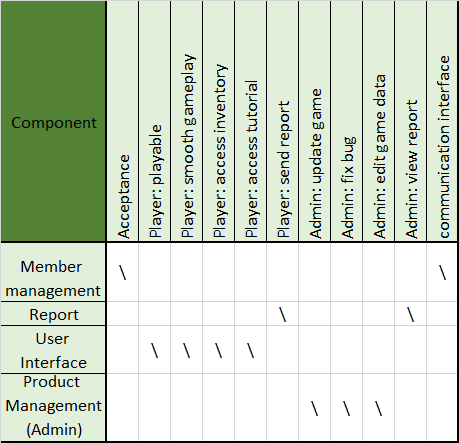
1. Admin will access to the database in game.

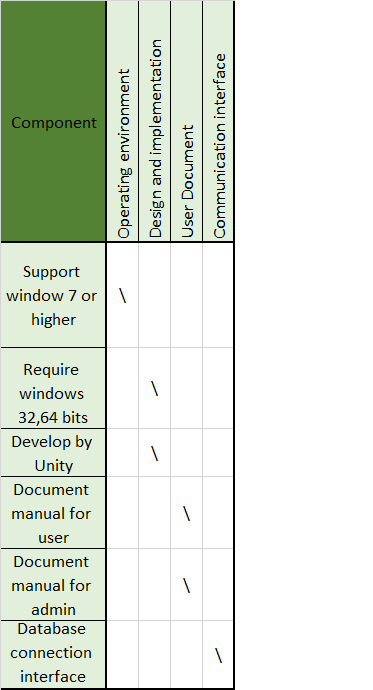
2. Admin add / delete any detail in program.

3. Admin will access to source code in the game.

4. Admin fix some bug and update code to make game updated.

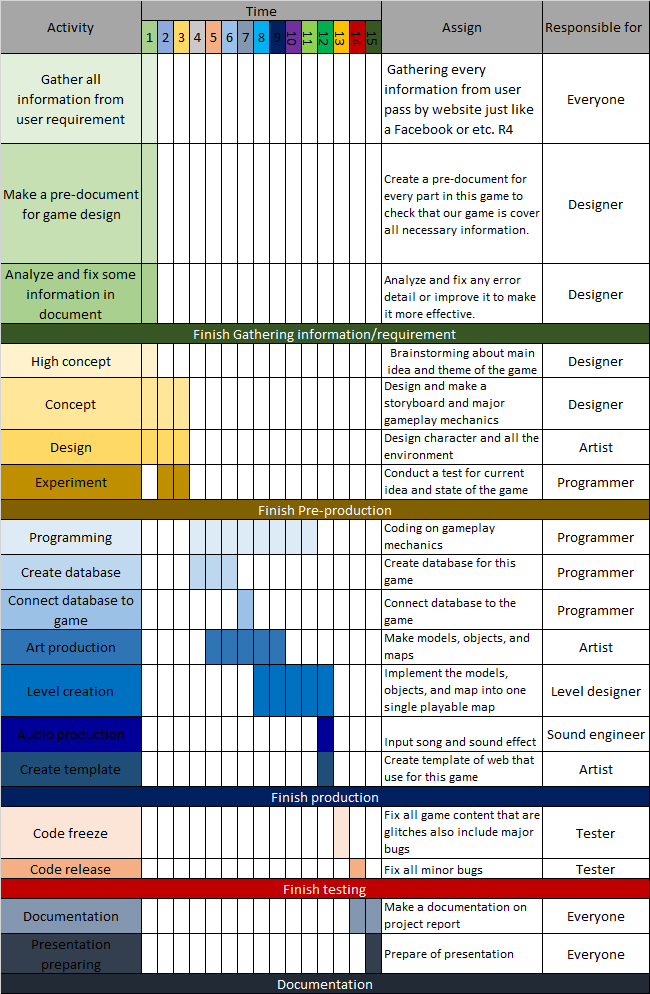
8. Requirement validation matrix





**Process manual specifications**

Project plan & monitoring method



Employee work/Task assignment process

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Task | | Hours | Start Date | End Date | Tester | Developer |
| 1 | Gather all information from user requirement | | 1 day | 1/2/2560 | 1/2/2560 | Thitiya | Everyone |
| 2 | Make a pre-document for game design | | 2 day | 2/2/2560 | 3/2/2560 | Thitiya | Designer |
| 3 | Analyze and fix some information in document | | 1 day | 4/2/2560 | 4/2/2560 | Thitiya | Designer |
|  | | Finish Gathering information/requirement | | | | | |
| 4 | High concept | | 2 day | 5/2/2560 | 6/2/2560 | Thitiya | Designer |
| 5 | Concept | | 10 day | 7/2/2560 | 16/2/2560 | Thitiya | Designer |
| 6 | Design | | 10 day | 10/2/2560 | 19/2/2560 | Thitiya | Artist |
| 7 | Experiment | | 8 day | 14/2/2560 | 21/2/2560 | Thitiya | Programmer |
|  | | Finish Pre-production | | | | | |
| 8 | Programming | | 50 day | 22/2/2560 | 13/4/2560 | Thitiya | Programmer |
| 9 | Create database | | 10 day | 26/2/2560 | 7/3/2560 | Thitiya | Programmer |
| 10 | Connect database to game | | 4 day | 7/3/2560 | 10/3/2560 | Thitiya | Programmer |
| 11 | Art production | | 30 day | 22/2/2560 | 23/3/2560 | Thitiya | Artist |
| 12 | Level creation | | 30 day | 13/3/2560 | 11/4/2560 | Thitiya | Level Designer |
| 13 | Audio production | | 3 day | 12/4/2560 | 14/4/2560 | Thitiya | Sound engineer |
| 14 | Create Template | | 4 day | 15/4/2560 | 18/4/2560 | Thitiya | Artist |
|  | | Finish Production | | | | | |
| 15 | Code freeze | | 7 day | 19/4/2560 | 25/4/2560 | Thitiya | Tester&Programmer |
| 16 | Code release | | 7 day | 26/4/2560 | 2/5/2560 | Thitiya | Tester&Programmer |
|  | | Finish Testing | | | | | |
| 17 | Documentation | | 10 day | 3/5/2560 | 12/5/2560 | Thitiya | Everyone |
| 18 | Presentation preparing | | 7 day | 10/5/2560 | 16/5/2560 | Thitiya | Everyone |
|  | | Documentation | | | | | |

**Name & Role of each member**

|  |  |
| --- | --- |
| Name | Role |
| Rittichai Sriubolmas | Art designer |
| Thitiya Trithipkaiwanpon | Tester, Level designer |
| Saranyu Pienpitak | Programmer |
| Tissatat Charoenchai | Designer, Sound engineer |

Final project cost method

Record hour from work

Record hours work form

Name: Thitiya Trithipkaiwanpon Employee No. 10001

Location: programmer

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Day | Start time | End time | Hours | Task | Update(%) |
| 1. | 1/2/25560 | 10.00 | 17.00 | 7 | Develop software | 10 |
| 2. | 2/2/25560 | 11.00 | 18.00 | 7 | Develop software | 20 |

Total hour 14 hours

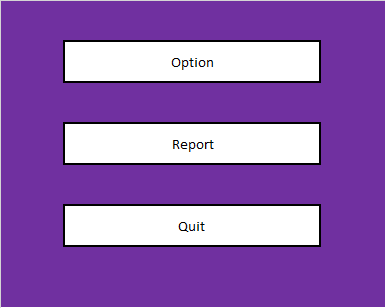
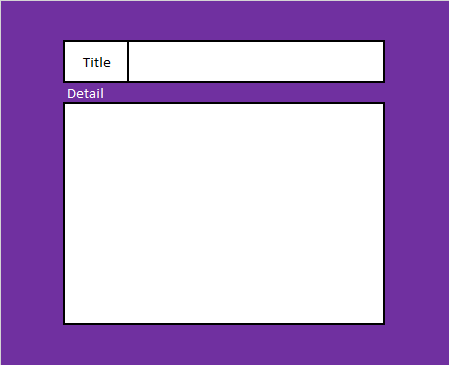
Payment and hour of employees

Payment and hours of employees

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Payment list | Bath Per hours | hours | Bath Per month |
| 1. | Artist | 300 | 30 | 9000 |
| 2. | Tester | 250 | 30 | 7500 |
| 3. | Level Designer | 400 | 40 | 16,000 |
| 4.. | Programmer | 400 | 50 | 20,000 |
| 5. | Designer | 500 | 50 | 25,000 |
| 6. | Level Designer | 350 | 40 | 14,000 |
| 7. | Other(Server,ETC) | - | - | 2000 |
|  | Total Price | - | - | 93,500 |

Requirement and change management process

This is a program that aim to entertain players and improve both sagacity and reflection skill of player. Developer will update a game by package from time to time. Player can report error of program in option system. Example of how roughly menu system is shown on picture below.



2 figures above show an outline of menu and report.

Software requirement form

Use to receive information of software form player. Player need to send information in report option by use formation above. Player can send any type of report such as: error, requirement and etc. Form the detail as follows:

|  |  |
| --- | --- |
| Report Information | |
| Title | Topic |
| Detail | Information |

System Diagram



Explanation of system diagram

**User:**

* Menu: this part is function that link to all option that player can do as following: option, report and exit.
* Report: is one of function in option that use to contract with administer.
* Check: this is a function use to check report that write correctly.
* Add: report that player send will save in database and waiting for administer to check and delete it.

**Administrator:**

* Check all: administer check all report that need to do or not.
* Fix error: administer fix and modify game system.
* Announce: administer announce user about update file.
* Upload: administer upload update file for user.
* CONFIGURATION MANAGEMENT PROCESS

At start of program, game will check itself that if the current user is administrator or not. If it’s administer, it will change to game-edit mode.

CONFIGURATION MANAGEMENT PROCESS

At start of program, game will check itself that I was administer or not. If it’s administer, it will change to game-edit mode.



MEASURES FOR SUCCESS

This process used to calculate how success of program. By calculate all factor that effect with a player as following: rating, bad and good report. Rating will collect for all player that complete the game or complete demo (program in process). All rating and report is the main measurement of program. Rating (1-5star)

Average of rating = ∑Rn /N

Rn = Rating of each player. N = number of rating

**Criteria for quality assessment**

Quality assessment of program will prove how good of program. By calculated all issue that effect with quality of program. The criteria is

Average of rating > 3 || less than 30 unique error report per month

Graph of recording

Average rating = (4.4+3.7+3.2+3.8)/4 = 3.775

Bad/Error report

January 29

February 22

March 11

April 7

From calculation above average rating, is 3.775, is more than criteria that we expected. Same as Bad/Error report that all 4 month have less than 30. So this program pass criteria for quality assessment.

USER ACCEPTANCE PROCESS

This process use to make sure the user accepts on developed program. Such as“user will not permission to modify program”, “agree with all privacy in program” and “accept all function in program”.



