โครงร่างวิทยานิพนธ์

(THESIS PROPOSAL)

ชื่อเรื่อง (ภาษาไทย) การใช้สถานะของเกมใน

ชื่อเรื่อง (ภาษาอังกฤษ) Implement Game State for Multi-Paths Quest in Structural Analysis Quest Generation

เสนอโดย นายธงธรรม จงมีสุข

เลขประจำตัว 5970176521

หลักสูตร วิศวกรรมศาสตร์มหาบัณฑิต

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะ วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

สถานที่ติดต่อ 189 โชคชัย 4 ซอย 9 วังทองหลาง วังทองหลาง

กรุงเทพฯ 10310

โทรศัพท์ 084 112 1819

อีเมล์ Thongtham.th.Chongmesuk@gmail.com

thongtham.c@student.chula.ac.th

อาจารย์ที่ปรึกษา ผศ.ดร.วิษณุ โคตรจรัส

คำสำคัญ (ภาษาไทย) ,,,,,เกมที่ใช้จริงในเชิงพาณิชย์

คำสำคัญ (ภาษาอังกฤษ) PROCEDURAL GENERATION, QUEST GENERATION, COMPUTER GAME QUEST, GAME STATE, COMMERCIAL GAME

โครงร่างวิทยานิพนธ์

**หัวข้อวิทยานิพนธ์**

ภาษาไทย

ภาษาอังกฤษ Implement Game State for Multi-Paths Quest in Structural Analysis Quest Generation

**1. ที่มาและความสำคัญของปัญหา**

INTRO: EXPLAIN GAME AND QUEST AND PROBLEM

Quest is one of the essential part of Role-playing Game (RPG game) genre. Quest will inform player about what have to be done for the quest/story to go forward. Quest also record actions the player had already done which bring the player to the current state of the quest/story. In a way, quest log is similar to the player’s own journal on the game world and how the player had interact with it so far.

However, many computer RPG games’ quest are too restrictive on what the player can do to move the quest/story forward. Many quests present player with limited path of actions player could do to complete the quest. This restriction in player freedom of action could lower player’s immersion when they realize that other players must also perform similar action to complete the quest, thus rendering the ‘my unique journal’ of player to be not so ‘unique’ compare to each other’s.

Some quests try to minimize this restriction by informing the player of what has to be done (goal state) without specific what action have to be taken, allowing player to discover the way to achieve the goal by themselves. However, in many case there are only 1 action or a set of actions that could lead to that specific state of the game. It could result in the same effect when the player discover that fact.

INTRO: EXPLAIN how PCG is used to deal with the problem

In order to create more variety and unique experience for each individual player, procedural quest generation (PQG) is developed and implement into the game. PQG is a subset of procedural content generation (PCG), a system which create content within game automatically and randomly (non-determined). The main purpose of PCG is to present each player with unique content that randomly change every time the game is played.

Still, current commercial PQG (in the game such as Elder Scroll Skyrim, Fallout 4) generated quest by combining multiple pre-scripted part of quest together to create a new quest. This, while guarantee the generated quest to always be functioning and consistent, still limit the flexibility of player action and quest content. Therefore new way to generate quest for computer game are being researched and developed. One such work is the structural analysis by Doran and Parberry [2011]. Doran and Parberry analysis quest from 4 MMORPG games and conclude common structure of quest which can be used in quest generating.

The structural analysis is a quest structure rule which can be used to construct quest.

INTRO: Previous work

In the work of Doran and Parberry [2011], their prototype quest generation use Prolog language to create all possible path to complete the input quest using the analysed structure / ‘grammar’. Then a path is selected as the ‘generated quest’ and present it to the player. Their quest structure / ‘grammar’ is further expand in Machado, Santos and Dias’ [2017] work.

Other works in quest generation that use different approaches include…..……….

Lee and Cho’s [2012] Dynamic Quest Plot Generation using Petri Net Planning.

Jens van de Water’s [2011]’s A Framework for Formalizing Dynamic Quests.

Most researches aim to create unique quests on the game environment and world that are pre-determined.

This may allow the system to generate unique quest for each player, but it cannot guarantee freedom of action of the player on how to complete the quest. ………………………..

INTRO: EXPLAIN how this thesis can improve it and solve the problem better

This thesis propose to implement game state checking and action resolve into the quest generation which use structural analysis. This system will replace the part of quest generation that generate list of task/action (path) player character has to perform to reach game state where the quest condition is complete. The new replacement system will allow the quest generation system to determine exactly how many paths the player can actually take to reach the same quest complete condition. The new ability to measure player freedom of action should allow the quest generation system to generate quest with higher flexibility without compromising the integrity of the generated quest.

The new system developed in this proposal aim to

ระบบการจัดการเนื้อเรื่องที่ได้พัฒนาขึ้นสำหรับวิทยานิพนธ์ฉบับนี้ มุ่งหวังให้เกิดองค์ความรู้ที่สามารถเผยแพร่ใช้กับเกมที่มีขายตามท้องตลาดได้จริง

The path consist can be summarized into “game state”(node) and “action”(path). Each action will take game state as input, and deliver modified game state as output. …………….

**2. ทฤษฎีที่เกี่ยวข้อง**

**2. Related Theories**

Related Theories in quest generation are consisted of [Role-playing Games], [] , [].

ทฤษฎีที่สำคัญในงานวิจัยนี้ประกอบด้วย รายละเอียดของเกมประเภทสวมบทบาท การจำแนกประเภทผู้เล่น และทฤษฎีการสร้างเนื้อเรื่อง

**2.1 Role-playing Games and Quest**

Role-playing Games (RPGs) are a genre in game that originated from pen and paper board game. In RPGs, player will control character (one or multiple) to explore the game world and interact with it. The main goal of RPGs is not to ‘win’ the game, but rather ‘interact’ with the game world and observe how the game world will react to player action.

The term RPGs can also be used to describe game which has a progressive development mechanic in player’s character ability and equipment; such as unlocking ability to fly, or upgrading weapon to perform higher damage.

RPGs main focus element is narrative, exploration, strategic planing, and deep character interaction, rather than combat or precision timing. However, other genre may implement RPGs element to create sub-genre. For example, Action-RPGs such as Dark Soul series which emphasis more on real-time combat, and Stretegy-RPGs such as Final Fantasy Tactics series and Crusader King series which emphasis on complex planing and play closer to chess.

The narrative of RPGs story can be dynamic or static base on game story design. However, most RPGs story will have a main storyline which the whole narrative revolve around. This could be ‘saving your kingdom from Alien invasion’, ‘seeking your missing parent’, or ‘revealing the mystery of a certain anomaly’.

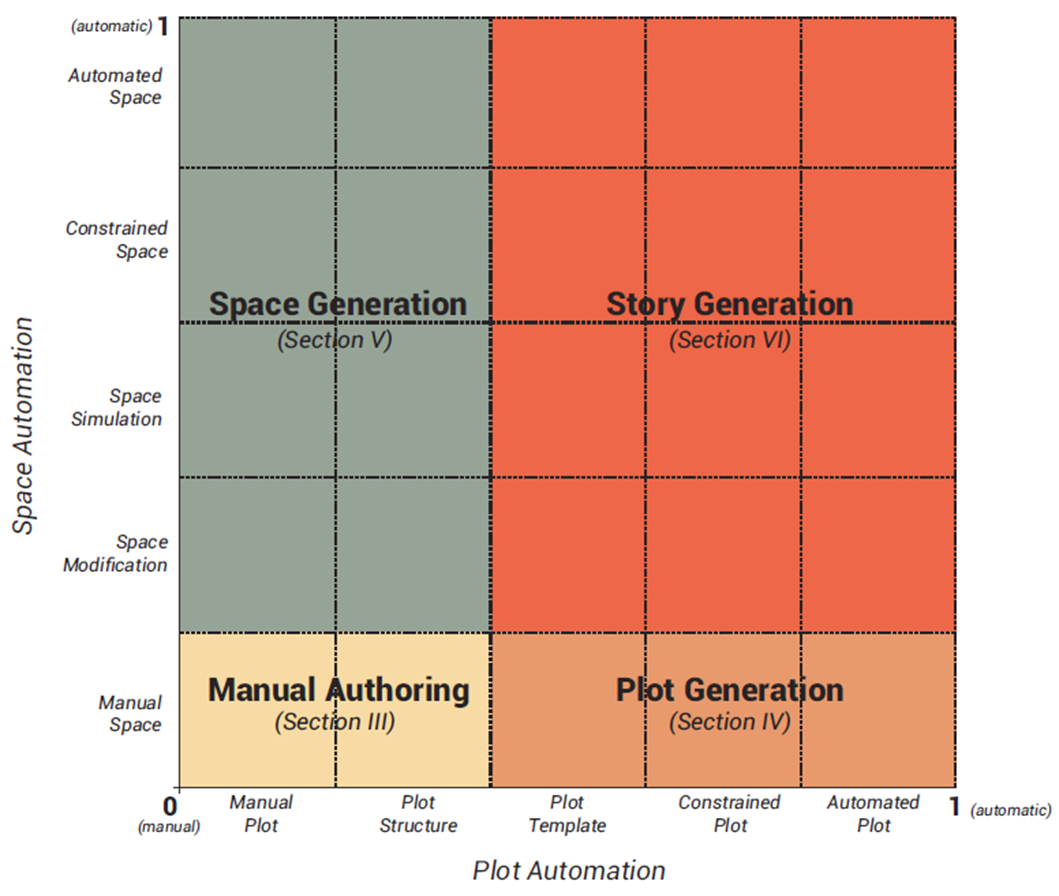
Quest is a task player character(s) can perform to receive reward range from money to the advance of the story. Quest can be categorized into 2 groups, main-quest and side-quest. Main-Quest is quest that revolve around the ‘main’ storyline of the game, while side-quest usually revolve around sub-plot of the story or something completely non-related to the story at all.

**2.2 Procedural Content Generation (PCG)**

Procedural Content Generation (PCG) is a system within a game which creates content for player consumption automatically as the player play the game. The content can be generated by selecting a static pre-determined object from a set, or combining multiple piece of objects together to create a more unique content, and so on. The content can range from new item, new enemy, new non-player character (NPC), new ability to new area. When a player opens a box that is designed to contain random weapon, and gets some weapon that is now a fixed weapon, is an example of PCG.

PCG is implemented into a game for multiple reasons. PCG allow the game to create a ‘unique’ experience for each individual player using the random nature of PCG, thus increase the replayability of the game and value of the game. PCG also reduce the burden of the game developer by lowering the amount of manual labour the developer has to perform to deliver the same amount of content when PCG is not used.

Procedural Quest Generation (PQG) is a subset of PCG and can be generalized into 2 categories, space and plot automation. Space is the game world object and environment, including item, NPC, geometry of the game world, weather, and such. Low space automation may result in only variety in enemy placement or randomness in reward from treasure chest. While high automation may generate a whole city with random NPC for the player to explore and perform task to complete a quest. Plot is the non-tangible part of content which dictate how the player and the tangible part interact with each other. Using a assassination event in a storyline as an example. Low plot automation may randomized only the place the murder happen, medium automation may vary the way the assassination is committed, but the assassination happen nonetheless. However, high automation, and high flexibility in a sense, may even result in an event that the victim actually survive and change the story that happen afterward.



The level of each type of automation, space and plot, can be used to identify the type of quest that could be generated from the PQG system. …………………….[talk about 4 type above picture]

|  |  |
| --- | --- |
| **Space Generation** | **Story Generation** |
| Neo Scavenger  Spelunky | Dwarf Fortress  Rimworld  Pokemon Mystery Dungeon |
| **Manual Authoring** | **Plot Generation** |
| The Witcher Series  Final Fantasy Series  World Of Warcraft | Mount&Blade Series  Sid Meier's Pirates  The Guild Series  The Elder Scroll:Skyrim  Fallout 3, Fallout: New Vegas, Fallout 4 |

[Also put picture of these game up there too.]

From the Figure XXX, Notable examples of PQG are “The Elder Scroll: Skyrim” and “Fallout series”, which use ‘Radiant AI’ from Bethesda Softworks. Both The Elder Scroll and Fallout have a static game world where most object are hand crafted and placed. The Radiant AI system is a PQG system which generate quest for player. The generated quest will have fixed task and narrative, but the object that the player have to interact to complete the quest will be randomly chosen from the object in the game world.………………

In order to create more variety and unique experience for each individual player, procedural quest generation (PQG) is developed and implement into the game. PQG is a subset of procedural content generation (PCG), a system which create content within game automatically and randomly (non-determined). The main purpose of PCG is to present each player with unique content that randomly change every time the game is played.

Still, current commercial PQG (in the game such as Elder Scroll Skyrim, Fallout 4) generated quest by combining multiple pre-scripted part of quest together to create a new quest. This, while guarantee the generated quest to always be functioning and consistent, still limit the flexibility of player action and quest content. Therefore new way to generate quest for computer game are being researched and developed. One such work is the structural analysis by Doran and Parberry [2011]. Doran and Parberry analysis quest from 4 MMORPG games and conclude common structure of quest which can be used in quest generating.

The structural analysis is a quest structure rule which can be used to construct quest.

**2.3 Structural analysis of quest**

Structural Analysis approach in quest generation is a way to construct quest in similar approach to constructing a sentence using ‘common’ grammar. The ‘grammar’ and ‘vocabulary’ rule of structural analysis was created by classification, analysing, and dissecting quests from multiple RPGs game to get a common pattern which all quest shared. In structural analysis approach, quests had been generalized into ‘motivation’, the distinct underlying drive (narrative) that compel the quest. Then within each ‘motivation’, the quest could be categorized into different ‘strategy’, the outline on how the quest (motivation) can be complete (satisfy). And finally, each ‘strategy’ could be linked to specific set of ‘Sequence of Actions’ which describe the general task (action) the player or NPC can to perform to complete the quest. The task (actions) are usually in the <ACTION> form, which can be **breakdowned** into specific ACTION depended on the **Action Rule** table.

**[Figure XAW] show example of <RuleSet> / Action Rule**

Figure XAW show an Action Rule Table from Doran and Parberry [2011],

**2.4 John Grey and Joanna Bryson’s Agent Interaction in Role-Playing-Games \* [CHANGE TITLE TO SOMETHING LIKE… “SIMULATING DYNAMIC NPC INTEREACTION” ]**

Quest are not exclusive to player only, NPC can also questing. Usually this kind of NPC-able Quest system are implemented in computer game to increase the level of dynamic environment and interaction of the game world.

……………….

\*This paper should be about using quest as a way for NPC to interact with each other.

\*EX. A hate B, so A got 'quest to kill B' and perform it.

……………….

A good example of player sharing quest pool with NPCs can be found in the original build of S.T.A.L.K.E.R, which is now known as S.T.A.L.K.E.R.: Shadow of Chernobyl. In that original build, every NPCs have the same set of available action similar to that of player, along with AI system that allow the NPCs to perform those action accordingly. This system was sound in when look into the game story, the player is one of the many S.T.A.L.K.E.R. in the area, and anyone can be ‘THE ONE’ who solve the mystery that litter around the game world. This system allow the player to encounter different ‘story’ depended on how the player and NPC chose to do what quests in what order. However, the GSC Game World (developer of S.T.A.L.K.E.R.) scrap the system because it is not a ‘fun’ experience for player. Testing players found that they were locked out of content and quests because other NPCs had already finished the quest. And most importantly, NPCs can finish the main story quest before the player and end the game prematurely. This is one of the problem when dynamic Questing NPCs system is used in game which has limited set of quests.

**2.5 A Parametric Analysis and Classification of Quests in MMORPGs**

G

**2.6 แบบจำลอง ความเชื่อ ความต้องการและเจตนา (Belief-Desire-Intention model or BDI)**

**3. Related Work**

Related work in quest generation are consisted of [Role-playing Games], [] , [].

**3.1 Doran and Parberry’s Structure Analysis [2011].**

Structural Analysis approach in quest generation is a way to construct quest in similar approach to constructing a sentence using ‘common’ grammar. The ‘grammar’ and ‘vocabulary’ rule of structural analysis was created by classification, analysing, and dissecting quests from multiple RPGs game to get a common pattern which all quest shared. In structural analysis approach, quests had been generalized into ‘motivation’, the distinct underlying drive (narrative) that compel the quest. Then within each ‘motivation’, the quest could be categorized into different ‘strategy’, the outline on how the quest (motivation) can be complete (satisfy). And finally, each ‘strategy’ could be linked to specific set of ‘Sequence of Actions’ which describe the general task (action) the player or NPC can to perform to complete the quest. The task (actions) are usually in the <ACTION> form, which can be **breakdowned** into specific ACTION depended on the **Action Rule** table.

**[Figure XAW] show example of <RuleSet> / Action Rule**

Figure XAW show an Action Rule Table from Doran and Parberry [2011],

**3.2 Quest Patterns for Story-Based Computer Games**

f

**3.3 Analysis of ReGEN as a Graph Rewriting System for Quest Generation**

f

**3.4 Hierarchical Generation of Dynamic and Nondeterministic Quests in Games**

f

**3.5 Natural Language Generation for descriptive texts in interactive games**

f

**3.6 Lee and Cho’s [2012] Dynamic Quest Plot Generation using Petri Net Planning.**

In this work, quest is defined as a sequence of event that happen to form narrative of the quest. the quest generation create quest by chaining multiple ‘events’ together.

**4. ด**

This thesis will use….

In term of programming, the system will be consisted of 2 languages.

1. Prolog. Prolog language is chosen because of its ability to query and backtrack.
2. Java.

----------

The system will be consisted of 3 main components.

1. Prolog query system
2. Eclipse java project which will generated quest structure from ruleset and store all possible path to complete the quest.
3. Ruleset which will dictate on how the quest will be generated and structured.

---------

The quest will be consisted of the following hierarchy

1. QuestFrame
2. Template
3. Component & RuleSet
4. GameState

MAKE SURE TO FIX <rule> AND RuleSet FIX that sometime got used wrongly and cause confusion

MAKE SURE TO FIX <rule> AND RuleSet FIX that sometime got used wrongly and cause confusion

MAKE SURE TO FIX <rule> AND RuleSet FIX that sometime got used wrongly and cause confusion

**[QuestFrame]:**

[QuestFrame] will be the main Framework of the quest. It will contain a List of Template and …..

**Template:**

Template can be described as an ‘arc’ of the quest. Each Template will contain component(s) which represent different states of the quest. A Template can be considered as a ‘mini quest’ if they are by themselves. **Restriction in the size of each Template is implemented to prevented overlong quest. The size can be considered as the number of Components within the Template.**

**Component:**

Component is the smallest part of the pre-determined quest structure. Each component will contain “Start State” and “Goal State”, and <RuleSet> according to the type of the Component. Start State can be described as the condition that the game world must satisfy at the point the component will be initiated as a part of quest. Goal State is the condition of the game world that must be satisfied for that part of the quest to be considered as complete and player can advance to the next part of the quest.

**Compare to Doran and Parberry [2011], Component can be consider as the [2011] RuleSet. Since the path/sequence of action by player which used to be determined by breaking down RuleSet is REPLACE by the new ‘loop’ system.**

**GameState:**

GameState is the state of the game world. GameState is the collection of multiple ‘Game World Condition’.

GameState used in Component can be categorized into 2 types.

1. Start State: Start State is the requirement condition that the game world must satisfy in order to **start the component**
2. Goal State: Goal State is the game state that when met, that section of the quest is considered completed and the next section can be started, advancing the quest.

GameState used in …………..………………………

………………………………

………………………………

………………………………

**Game World Condition:**

An individual condition or happening in the game world. All Game World Condition in the current game world will be called “GameState”.

---------------------------------------

**NOTE: BELOW RULESET BREAKDOWN IS DIFFERENT FROM THE ORIGINAL SINCE THE new stystem ‘loop’.**

**Original Ruleset:**

Ruleset will be used to determine how <atom> will be generated from <rule>. This system will use the Ruleset from Machado, Santos and Dias’ [2017] work which extended from Doran and Parberry [2011].

**When <rule> is break downed, it should breakdown according to the RuleSet that dictate how <atom action> can be generated from <rule>. For the generated quest to have consistence task and content that fit the narrative of the quest, Doran and Parberry [2011] analysed multiple quests from 4 MMORPGs game to ………………… FIX THIS**

Each <atom action> can be descript as ‘task’ / ‘action’ that the player have to perform to advance the quest. Each <atom action> can be summarized to have ‘input Game State’ and ‘output Game State’. **As shown in Figure AXXA**

**[Figure AXXA]**

When the quest is generated in the previous system, Prolog will be used to breakdown all possible path / sequence of <atom action> from the initial <rule>. Then a path will be chosen and delivered as a quest. This path / sequence can be described as the sequence of <atom action> that could change the starting Game State to desired Game State that satisfy the quest goal. When the player perform the action descript within the <atom action>, the current Game State (input Game State) will change to ‘output Game State’ and so on until the Game State that satisfy the quest ending condition is met.

**NEW RULESET and Usage:**

In this system, the breaking down of <rule> into <atom action> using RuleSet is obsolete by the new system that can generated all possible path / sequence of actions that the player can take to advance from the initial Game State (‘input Game State’ in this case) to the desired Game State (‘output Game State’ in this case).

The new usage of the RuleSet is to prevent the ‘loop’ from achieving certain Game State status before another one that would result in **conflicting** quest, such as reporting the capture of NPC N before NPC N is actually capture.

To achieve this, the new usage of RuleSet will be to generate a sequence of Game State rather than a sequence of action. And instead of generating the sequence from start to finish (Prolog from root to all leafs), the new RuleSet will be used to generated these Game State from finish (goal Game State) to the starting Game State. **For example, <steal> can be broken down into [<goto> stealth take] or [<goto> <kill> take] where the <Rule> can be expanded further.**

**EACH COMPONENT CAN BE CONSIDER AS <Atom Action>**

**INITIAL COMPONENT = <RULE>**

**BREAKDOWN COMPONENT = <ACTION>**

When the <rule> is breakdown in this system from the Template and Component according to RuleSet, the ‘output Game State’ of the last <atom action> will be account for. The system will look at the last <atom action> and determined what’s the ‘output Game State’ of that last <atom action> could be, then determine what the ‘input Game State’ should be so that the last <atom action>’s ‘output game State’ is possible. For example, if the <atom action> is <deliver item A1 to NPC 2B>; then the ‘output Game State’ of this <atom action> is “NPC 2B possess item A1”. Thus, the ‘input Game State’ of thie <atom action> should be “NPC 2B NOT possess item A1”.**As shown in Figure AXXA**

**[Figure AXXA]** (figure of whole ruleset, last one analysis “start” condition)

Then when the next <atom action> is analysed (2nd last <atom action> in this case), the Game State “NPC 2B NOT possess item A1” will be passed down as an additional Game State. **As shown in Figure AXXA**

**[Figure AXXA]** (figure of whole ruleset, 2nd last one analysis “end” condition / AKA: condition required to be met after the <action> is done)

After the analysis , the result would be….. **As shown in Figure AXXA**

**[Figure AXXA]** (figure of whole ruleset, 2nd last one analysis “start” condition / AKA: condition required to be met before the <action> canstart)

When the process exhaust all <atom action> and reach the first ‘input Game State’ of the first <atom action>, the collected Game World Condition (called **Restriction State** from now on) will be used to prevent the system from choosing conflicting action between each Game State. Now when the system start to generate path from the root Game State (start State of the first Component) to the goal state of the first component, the system will check if the Game State Condition generated by these paths conflict with the Restriction State.

For example,

**As shown in Figure AXXA-AXXZ**

**[Figure AXXA]** (figure of [step by step on how the **Restriction State** cancel out path that conflict] 1st step.

**-**

**[Figure AXXZ]** (figure of [step by step] last step.

By doing this, the system can prevent conflict and inconsistence sequence of action. The result path from each Game State to next Game State would retain the consistence of the original RuleSet, while flexible enough to take advantage of the new path generation system.

**NEW RULESET and Usage 2:**

On the other hand, the system can disable RuleSet completely when generating quest path. <RuleSet> that is yet to be broken down into <Atom Action> in the Component can be ignored so that the system would have less restriction in the creating of path.

In this type of configuration, only the ‘Start State’ of the Component will be ……

Similar configuration (disable of using any **<RuelSet> to create Restriction State**) can also be used at Template level……….

**\*\*\*CURRENT CONFIG:**

**Restrucition State will be apply only in component that were created in the same template only.**

**When the RULESET reach the start of the template, and will go into the end of the previous template, all Restrucition State will be reset.**

**THIS SETTING CAN CHANGE TO SEE HOW IT EFFECT THE QUSET GENERATION, BUT FOR DEFAULT SETTING THIS WILL DO.**

***RuleSet ‘breakdown’ within Components***

1. ***According to the <RuleSet> from [2017]***
2. ***According to the***

***RuleSet ‘breakdown’ between Components in the same Template***

***f***

***RuleSet ‘breakdown’ between Components in the different Template***

***f***

**Since there’s a rule governing the connection between Component between these elements already, theese instruction is redundant. All component can be viewed as a long list of Component form single template, no matter how many template they actually generated from.**

This is further restricted in the original RuleSet breakdown by manually selecting <RuleSet> in each “quest section” (Template in this case). The <RuleSet> are put in certain order and type to make sure that no messy or convoluted set of <atom action> is generated.

However, with the new ‘path generation’ of this system, such manual restriction may not serve the system well compare to the previous system. Therefore, it may be more beneficial

**Methodology**

Linking between Template

Each type of [QuestFrame] will have pre-determined set of Template. These are manual

Linking between Component within same Template

1. 1st Component’s Goal State must NOT conflict with 2nd Component’s Start State.
2. 1st Component’s Goal State SHOULD match with 2nd Component’s Start State. But not MANDATORY

Linking between Component across Template

1. 1st Template last Component’s Goal State must NOT conflict with 2nd Template first Component’s Start State.

Quest Generation

[Explain from JAVA]

When the system generates a quest. It will start by initiating a [QuestFrame].

Then template(s) will be selected according to the desired quest type. The [QuestFrame] can contain multiple templates within itself. The number of template is determined by the initial configuration of that quest. Short quest may contain only 1 template, while longer and more complex quest can contain multiple templates linked together.

Then within each template, component(s) will be created according to the selected template. The system will consult with the template table that store all templates’ components setting. The <Component> will then be broken down into Component. (ต้องกลับไปเขียนอธิบายถึงความต่อระหว่าง <Component> and Component)

**[Figure AQW] this is figure that show the current status of the [QuestFrame] and quest after all above step is done**

**[Figure AQW] this is figure that show the current status of the <Component> is breakdown into Component**

After the component(s) are created and breakdown, the quest outline can be considered complete. The next step is to fill in detail, narrative, and specific condition of the quest for it to be playable. Due to the changing in the system, RuleSet table no longer represent the direct task that the player has to perform, but rather the table which dictate how GameState within the quest will hop from one to the next.

First, the starting condition of the quest has to be created. The starting GameState will be set up using the **Restriction State** and other.……………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...………...

In order to obtain **Restriction State,** allcomponents within the quest will have their StartState and GoalState breakdown into a **list** of GameState from all components (AKA NEW <RULE>). Then the system will start to read the list from **back to front, while assembling the necessary GameState that must be met if the GameState is to be achievable from GameState that come before. [เขียนแก้ให้เข้าใจได้ง่ายขึ้น]**

After the **Restriction State** is created, the **[quest generating system]** will be able to determine what condition is prohibited during the ‘path’ generating in order to not create a conflicting and inconsistent action path.

…………………………………..

[Continue Explain to JAVA]

---------

**Expected Outcome**

The system can generate quest according to RuleSets and their conditions.

The generated quest can be completed by player character.

The generated quest can be

T

---------

**Measurement**

Using ReGen

Each ‘path’ / sequence of actions the player can perform to complete a quest

**5. Objective**

This thesis

**5. วัตถุประสงค์ของการวิจัย**

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาระบบการดำเนินเนื้อเรื่องของเกมประเภทสวมบทบาทที่สามารถปรับเปลี่ยนเนื้อเรื่องตามบุคลิกลักษณะการเล่นของผู้เล่นเพื่อให้เนื้อเรื่องมีความเหมาะสมต่อผู้เล่นและสามารถทำให้ผู้เล่นพึงพอใจได้

**5. Scope of Work**

This thesis