Збірна команда Закарпаття «Smart bears»

Технічна документація
до стратегічної гри «Uzhlyandian Wars»

SMART BEARS

Закарпатська область , 2016 рік

ЗМІСТ

1. Про гру «Uzhlyandian Wars»

Ігрова індустрія в останні роки зазнає великого прогресу. Щороку створюються мільйони нових ігор різних жанрів. Одним із найстаріших жанрів є стратегії, хоча вони не втрачають популярності й до тепер. «Uzhlyandian Wars» представляє гру в цьому жанрі. Вона має зручний геймплей, чудову графіку та багато можливостей. Тут є 3 основні режими гри з різною складністю та особливостями. Кожен з них буде описаний в наступних розділах. Також, для кожного режиму було створено штучний інтелект 3-х рівнів складності. Якщо ж користувач хоче розважитись з друзями, то тут є мультиплеєр для двох гравців. Гра виконана в історичній тематиці і сподобається будь-якому фанату. Ви зможете відчути себе вождем народу і зіграти за таких великих людей, як Наполеон, Сталін , Вашингтон та інших історичних персонажів.

2. Рекомендовані системні вимоги:

- Операційна система : Windows XP/7/8/10
- Процесор: 1 ГГц
- Оперативна пам'ять: 1 Гбайт
- Відеокарта: ATI Radeon 9400 128 Мбайт
- Аудіокарта: DirectX-сумісна
- DirectX: 9.0c
- Твердий диск: 4 Гбайт вільного простору

3. Інструменти розробників:

3.1. Adobe Photoshop — графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в області комерційних засобів редагування растрових зображень, і найвідомішим продуктом фірми Adobe.

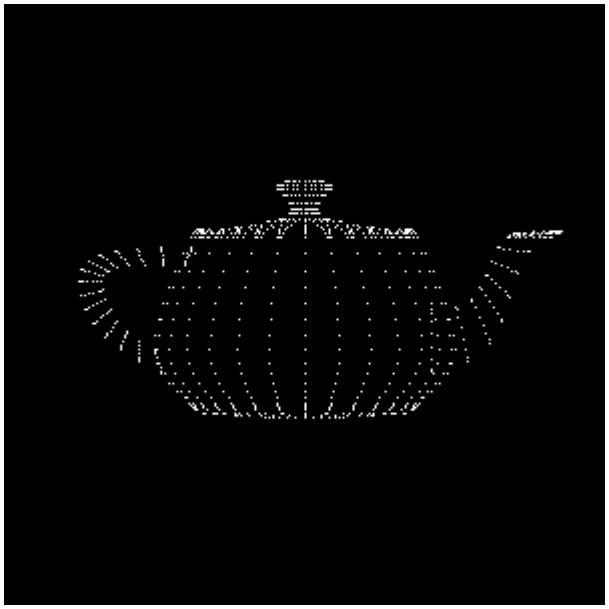
3.2. Набір компіля́торів GNU — набір компіляторів для різних мов програмування. GCC — вільне програмне забезпечення, розроблене Фондом Вільних Програм під ліцензією GNU GPL та GNU LGPL, і є ключовою складовою набору знарядь розробки GNU (GNU development toolchain).

3.3. Code::Blocks — вільне кросплатформенне середовище розробки програмного забезпечення. Code::Blocks написана на C++ і Використовує бібліотеку wxWidgets. Маючи відкриту архітектуру, може маштабуватись за рахунок додаткових модулів. Підтримує мови програмування C, C++, D та Fortran.

3.4. Бібліотека OpenGL — специфікація, що визначає крос-платформовий API для написання застосунків, що використовують комп'ютерну графіку.

Документація : https://www.opengl.org/documentation/

3.5. GLUT — бібліотека утиліт для застосунків під OpenGL, яка в основному відповідає за системний рівень операцій вводу-виводу при роботі з операційною системою і включає функції: створення вікна, моніторинг за вводом з клавіатури і подіями комп'ютерної миші; функції рисування ряду геометричних примітивів: куб, сфера, чайник та інші.



Документація : https://www.opengl.org/resources/libraries/glut/spec3/spec3.html

3.6. DevIL — кросплатфрмова програмна бібліотека, ціллю якої є створення спільного інтерфейсу програмування застосунків для різних файлових форматів графічних зображень.

Документація : http://openil.sourceforge.net/docs/

4. Правила гри

4.1. Гра для двох гравців «Uzhlyandian Wars»

відбувається на полі, поділеному на шестикутні ділянки. Кожен гравець мешкає в місті, що має підконтрольну територію. Виграє той, чия територія на кінець гри буде більшою.

Гравці ходять по черзі. Для збільшення території гравець повинен будувати фортеці, які можна встановлювати або на власній ділянці, або на ділянці, суміжній з власною, якщо там не встановлена інша фортеця. Після цього дана клітинка та всі суміжні з нею стають вашими. Виключенням з цього правила є тільки місто – його захопити неможливо.

4.2. Режими гри

4.2.1. “I’m too young to die”

У цьому режимі гравці грають на прямокутному полі, де всі клітинки придатні для будівництва. Найпростіший варіант гри, саме він рекомендується новачкам для освоєння базових навичок гри.

4.2.2. “Hey, not too rough!”

Цей режим відрізняється від попереднього змінами у рельєфі місцевості, у якому можуть з’являтися клітинки з горами і водою. Будувати на них не можна, але їх можна захопити для збільшення підконтрольної території.

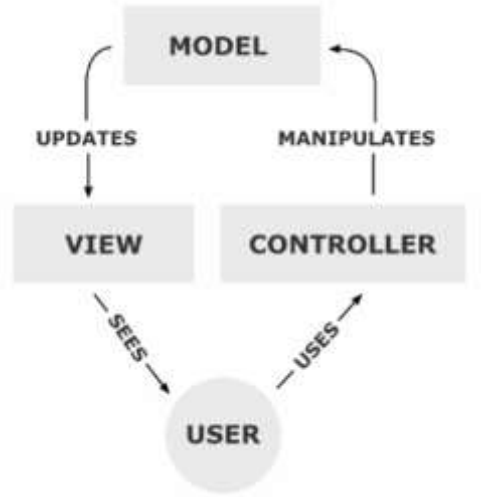
4.2.3. “Hurt me plenty”

Цей режим відрізняє від попереднього появою нового гравця на полі – генерала. Тепер право будувати фортеці належить лише їм. За один хід вони можуть або переміститись на 2 клітинки, або переміститись на 1 і побудувати в ній фортецю, або побудувати фортецю в тій клітинці, де він стоїть. Після встановлення фортеці генерал зникає.

5. Вихідний код

Всі матеріали по проекту можна знайти на git-репозиторії за адресою : https://github.com/GiraffeHarold/Uzhlyandian-Wars

6. Структура проекту



Наш проект побудований за схемою MVC(Model-View-Controller), яка передбачає поділ програми на 3 незалежні модулі. До обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель.

7. Файли та їх призначення

- **Компонент Model**
  - Basic_classes.cpp, basic_classes.h – файли, що визначають базові класи і способи зберігання даних.
  - Generals.cpp, generals.h – файли, що відповідає за основні операції з генералами.
  - Intellect.cpp, intellect.h – файли з процедурами інтелекту

- Levels_intellect.cpp, levels_intellect.h – файли, що реалізовують різнорівневий інтелект.
- Map_generator.cpp, map_generator.h – файли, що відповідають за генерацію випадкових карт.
- Loading_game.cpp, loading_game.h, load_data.txt – файли, що відповідають за завантаження збережених ігор.
- Save_game.cpp, save_game.h– файли, що відповідають за збереження ігор.
- Global_vars.h – файли, що відповідають за опис змінних, що використовуються в декількох файлах.

- **Компонент View**
  - Textures.cpp, textures.h – файли, що відповідають за загруження текстур.
  - Draw_menu.cpp, draw_menu.h – файли, що відповідають за виведення меню.
  - Drawing.cpp, drawing.h – файли що відповідають за виведення всіх текстур крім меню.
  - End_game.cpp,end_game.h – файли що відповідають за виведення результатів гри.
- **Компонент Controller**
  - Menu_mouse.cpp, menu_mouse.h , Menu_mouse.cpp.save-failed – файли, що відповідають за роботу головного меню.
  - Pause_menu.cpp, pause_menu.h – файли, що відповідають за роботу внутрішньоігрового меню.
  - Intelect.cpp, intellect.h – файли, що відповідають за роботу інтелекту.
  - Button.cpp, button.h – файли, що відповідають за роботи конпок.
  - Controller.cpp, controller.h – файли, що відповідають за обробку ходів гравця.
  - Generals.cpp,generals.h – файли, що відповідають за обробку ходів генералів.
  - Log.txt – текстовий файл, з якого можна зчитувати інформацію про ходи учасника.

- **Звязуючі файли**
  - All_includes.h – файли, що відповідають за підключення всіх бібліотек і всі дефайни.
  - Project.cbp, project.depend, project.layout – файли проекту, призначені для розробників, у яких зберігається інформація про файли проекту, стан роботи, зв’язуючі файли
  - Main.cpp – основний файл запуску.

8. Алгоритмічне наповнення проекту

8.1. Генератор карт

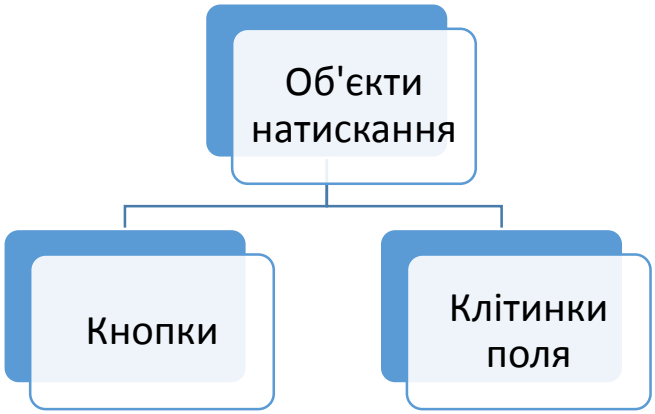
Спеціально для 2-го та 3-го режиму гри було розроблено генератор рельєфу місцевості, який генерує випадкову карту за такими параметрами :

- Розміри Маленька(5x5 - 8x8)
  Середня(10x10-20x20)
  Велика(20x20-35x35)
  Гігантська(35x35-50x50)
- Тип
  Материк – суцільна суша із невеликою часткою водного рельєфу
  Континенти - два материки, розділені морем, міста суперників на різних материках
  Острови – переважно водний рельєф, з невеликими ділянками суші(рекомендовано для 3 режиму)
- Гористість
  Гірський масив – переважно гірська місцевість
  Передгір'я – значна частина гірського рельєфу
  Рівнина – частка гір мізерна або вони відсутні

Окрім цього, в цій грі є можливість зіграти не на випадковій карті, а на заздалегідь обраній із даного списку згенерованих карт, серед яких є карти з унікальним і неповторним ландшафтом, карти з нестандартними ситуаціями тощо.

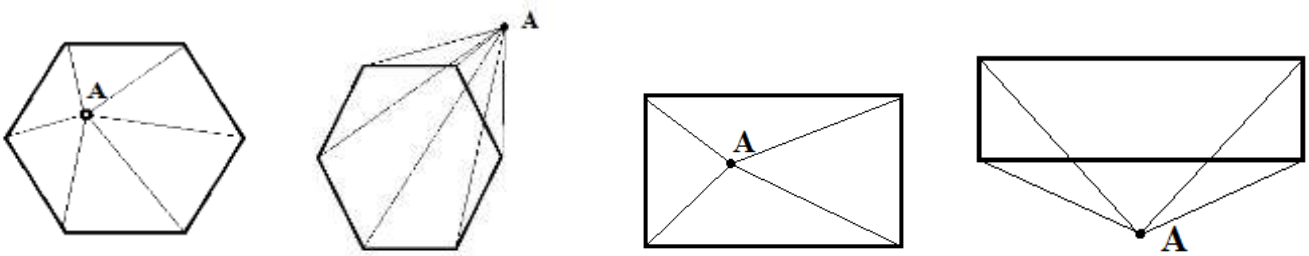
8.2. Реакція на натискання кнопки

У нашому проекті є 2 основні об’єкти, що реагують на натискання на них : кнопка і клітинка поля.



Якщо представити ці об’єкти як геометричні фігури(кнопка – прямокутник, а клітинка - шестикутник), то ми зведемо цю задачу до задачі на знаходження многокутника, в якому знаходиться точка.

Для цього нам потрібно уміти визначати, чи знаходиться точка в многокутнику. Якщо точка знаходиться в многокутнику, то сума площ трикутників, утворених точкою і кожною зі сторін, має бути рівною площі многокутника.



Формула для обрахунку площі многокутника виглядає так :

$$S(A) = \frac{1}{2}\sum_{i=1}^{n}(A_i.x - A_{i-1}.x) \times (A_i.y + A_{i-1}.y)$$

Твердження, яке виконується, якщо точка належить многокутнику :

$$F(p, A) : S(A) = \sum_{i=1}^{n} S(p, A_i, A_{i-1})$$

Тоді функція, яка визначає, в якому многокутнику знаходиться точка виглядатиме так :

$$GetPolygon(p) = A, \text{при } A \in Polygons \text{ і } F(p, A)$$

Також виконана

8.3. Штучний інтелект для 1-го та 2-го режимів

Введемо поняття пріоритету клітинки – числової константи, яка залежить від того, наскільки нам вигідно захоплювати цю клітинку.

$$\boldsymbol{priority}(\boldsymbol{p}) = \begin{cases} \mathbf{60}, & \text{якщо } \boldsymbol{p.relief} = \boldsymbol{EMPTY} \\ \text{інакше } \mathbf{5} \end{cases}$$

Введемо таке поняття як потенціал клітинки – дійсне число, яке показує , у якій мірі ця клітинка збільшить наші шанси на виграш. Очевидно, що чим більший потенціал, тим кращим для нас цей хід буде. Формула для обрахунку потенціалу клітинки *p* виглядає так :

$$\boldsymbol{R}(\boldsymbol{p}) = \sum_{\boldsymbol{to} \in \boldsymbol{Ways}(\boldsymbol{p})} \boldsymbol{priority}(\boldsymbol{to}) * [\boldsymbol{p.color} \neq \boldsymbol{to.color}]$$

, де $\boldsymbol{Ways}(\boldsymbol{p})$ – функція, що визначає множину суміжних клітинок для клітинки $\boldsymbol{p}$.

8.3.1. Режим Easy

Режим Easy не вимагає від інтелекту ідеальної гри, навпаки, він повинен грати так, щоб навіть найслабший гравець міг виграти. Тому в режимі Easy інтелект закономірно обирає найлівішу клітинку, доступну для ходу. Це було зроблено для того, щоб гравець, який зрозуміє цю закономірність, міг спрогнозувати поведінку інтелекту і , можливо, виграти його передчасно, заблокувавши йому всі ходи.

Свідомо можуть ігноруватися клітинки з найкращим потенціалом, адже тоді у інтелекту з’явиться можливість виграти.

8.3.2. Режим Medium

У цьому режимі суть алгоритму полягає у повному переборі всіх можливих варіантів ходу і обрання із них не найкращого, а середнього за значенням потенціалу. Під час вибору допускається незначне відхилення в гіршу сторону, щоб було важче спрогнозувати хід суперника.

8.3.3. Режим Hard

В режимі Hard інтелект обирає варіант з найкращим потенціалом і ставить клітинку туди, де значення потенціалу є найбільшим. Оскільки потенціали не завжди можуть точно спрогнозувати поведінку іншого гравця, то використовується функція , яке робить незначні відхилення у обрахунку, які сприятливо впливають на гру інтелекту.

8.4. Штучний інтелект для 3-го режиму

8.4.1. Режим Easy

Інтелект для цього режиму мало чим відрізняється від інтелекту для 1-го і 2-го режимів. Генерали завжди намагаються йти в найлівішу доступну клітинку і якщо хід можливий то завжди ходить. Цю закономірність легко помітити і передбачити ходи супротивника. В такому випадку не важко навіть заблокувати інтелект (зробити так, щоб він не мав куди ходити) і змусити його здатись.

8.4.2. Режим Medium

В режимі Medium інтелект діє жадібно, кожного разу намагаючись забрати клітинку з найменшим потенціалом. Потенціал клітинки оцінюється як сума відстаней від неї до всіх інших генералів супротивника.

$$\boldsymbol{priority}(\boldsymbol{p}) = \sum_{\boldsymbol{gen} \in \boldsymbol{opponent\_generals}} \boldsymbol{dist}(\boldsymbol{p}, \boldsymbol{gen})$$

, де $\boldsymbol{dist}(\boldsymbol{p}, \boldsymbol{gen})$ **–** відстань від клітинки $\boldsymbol{p}$ до генерала ***gen.***

Таким чином вони будуть якомога ближче до генералів суперника. Фортеці інтелект намагається будувати тільки на останньому кроці і захоплювати максимальну кількість території, яку раніше захопив суперник.

8.4.3. Режим Hard

Для цього режиму основним завданням є не тільки захопити чим більшу територію, а й при можливості оточити замок суперника, забравши в нього можливість зробити

наступний хід. Для цієї мети всіх генералів було поділено на три типи:

- Builder- повинен ставати в такі місця, щоб забрати якомога більше території, побудувавши фортецю на останньому кроці. Вони намагаються триматись на відстані 2 шестикутників один від одного, щоб не забирати при побудові одну й ту ж територію.
- Attacker- повинен йти до замку ворога і намагатись оточити його. Якщо навколо замку гравця перебуває 6 таких генералів, то він не зможе вивести з нього свого генерала, а отже і закінчити хід, що змушує його здатись.
- Defender- повинен залишатись біля нашого замку, щоб ворог не зміг повністю заблокувати з нього вихід.

Залежно від рельєфу, позицій суперника та особливостей карти генерали можуть міняти свої властивості та стратегію.