УчИМИ Ученически институт по математика и информатика

ШЕСТНАДЕСЕТА УЧЕНИЧЕСКА СЕКЦИЯ
УС'16

ТЕМА НА ПРОЕКТА

Into The Woods

Автор:
Атанас Великов Орманов
ПМГ „Иван Вазов“, Димитровград, XI клас

Научен ръководител (консултант):
Петър Петров, програмист, САП Лабс България

Into The Woods

Nowadays games have become the main source of entertainment. “Into The Woods” is exactly that type of game. Its aim is to entertain and even amuse the player with its twisted sense of humor and at the same time to give him a hard time with its randomly generated levels. The procedural level generation is at the base of the game to give the player a unique experience by making him play a unique playtrough every time. The game concept is based on the fairy tales fairy tales of the Brothers Grimm, Charles Parrot and other authors.

В гората

В днешно време игрите са се превърнали в основен източник на развлечение. Точно такава игра е „Into The Woods“. Целта й е да забавлява играча с изкривеното си чувство за хумор и в същото време да го затруднява с генерирани на случаен принцип нива. Това генериране на случаен принцип ще стои в основата на играта, за да може всеки път когато някой играе играта, да играе в една напълно уникална нейна конструкция. Концепцията на играта е базирана върху приказките на Братя Грим, Шарл Перо и други автори.

• Увод

В днешно време видеоигрите са се превърнали в основен източник на развлечение. С времето тази интерактивна медийна форма става все по-впечатляваща. Като човек, който ги използва и в същото време проявява интерес в разработването им, аз започнах разработката на една изцяло развлекателна игра. Целта ми е да създам игра с една оригинална концепция, която ще придостави на потребителите си едно уникално изживяване.

• Изложение

Into The Woods („В гората“) е 2D развлекателна игра, характеризираща се частично като:

- Action – защото в играта ще се разиграват битки с разни същества
- Roguelike – заради изгледът отгоре, процедурното генериране на нивата, дизайнът разделен на квадрати и елемента permadeath (когато играчът загуби играта започва изцяло отначало)
- Dungeon Crawler – сценарий, който включва героят да е затворен в лабиринт, тъмница, гора или в друга такава среда. Проправяйки си път през всичките враждебни същества, той граби съкровища и предмети, които му помагат/пречат в играта
- Fantasy – заради фантастичните елементи, същества и герои, които включва концепцията

Концепцията на играта е базирана главно върху някои от приказките на Братя Грим, Шарл Перо и др. Тези приказки ще са представени като първите си и оригинални версии, които включват странни, мрачни, гротески и социопатични елементи. Повечето същества (врагове) ще представляват плашещи същества или герои извадени директно от тези „детски“ приказки. Именно заради тази мрачна обстановка, която планирам да създам играта ще има възрастово ограничение 16+. На пръв поглед може да звучи леко отблъскващо, но красотата в подобна игра е в цялостта на създадения свят и представянето на цялата концепция в анимационен стил.

Вдъхновение за започването на тази разработка получих от играта The Binding of Isaac, която има подобен стил на игра и също така изкривено чувство за хумор. Други подобни игри на които се възхищавам са Hammerwatch, Hero Siege, Our Darker Purpose.

За разработването на проекта съм използвал програмите Unity3D – за механики и цялостно оформление, и Adobe Photoshop CS6 – за графичен дизайн.

Основните неща които съм направил до момента по играта, са:

- възможност за движение в общо 8 посоки
- възможност за стрелба в общо 8 посоки
- параметри на героя – живот(health), енергия(energy), скорост(speed), обсег(range), честота на стрелба(attack speed) и щета(damage)
- едно уникално оръжие
- механики за живота и енергията
- алгоритъм за намиране на най-кратък път между две точки(pathfinding)
- едно статично чудовище и едно динамично(използва pathfinding)
- генериране на верига от елементи на случаен принцип
- генериране на пещери на псевдослучаен принцип, които да репрезентират елементите споменати в горната точка
- графичен дизайн за някой обекти

- **Основни етапи в реализирането на проекта**

- Написвам програмния код
- Създавам графичните обекти и ги имплементирам в Unity
- Сглобявам игрови обекти от отделни елементи (скриптове, графични обекти, физични елементи и др.)
- Оформям игрова сцена от различните игрови обекти
- Подреждам сцените в определена последователност

- **Ниво на сложност на проекта**

Основна трудност в създаването на играта е преодоляването на обема на съдържанието:

- създаване на преплитащ се сюжет
- много вариации на AI (изкуствен интелект)
- сглобяване на много механики и алгоритми в една добре работеща игра
- авторски графичен дизайн

- **Функционално развитие**

Играта е съставена от следните части:

- графична – тази част включва единични Sprite обекти и Sprite анимации, като графичният дизайн ще е изцяло в 64 пикселов формат
- програмна – тази част включва програмиране на езика C#
- game engine – в тази част се сглобява всичко, за да се получи крайният продукт

а) Верига от елементи

За създаването на веригата от елементи съм използвал авторски алгоритъм, който работи по следния начин:

Първо се създава матрица, изградена от квадрати, които ще наричам *клетки*. Чрез индексите на този двумерен масив се намират съседните на всяка клетка и се добавят към списък. Така всяка клетка има списък със своите съседни. Всяка клетка, която участва във веригата, ще бъде добавяна към списък, който ще нарека N, а клетките, които бъдат добавяни, ще бележа с X. Първата клетка, която е добавена към N, е тази в центъра на матрицата. (Табл. 1)

			X			

Табл. 1

Следващата избрана клетка ще бъде избрана на случаен принцип от списъка със съседни на X. След това двете клетки ще се изключат взаимно от списъците си, за да не могат да бъдат избрани отново. (Табл. 2)

			X1			
			X			

Табл.2

Сега е избрана на случаен принцип една от двете клетки, участващи в N, и се повтарят предходните действия. При процедирането на алгоритъма обаче трябва да бъде спазено едно условие – клетката, която бъде избрана да се добави към N, не трябва да има 2 или по-малко елемента в списъка си със съседни. Така достигайки определен брой елементи, участващи в N, алгоритъмът спира и се получава верига от клетки. (Табл. 3)

	X9	X4		X7		
		X3	X1	X2	X6	
			X			
		X8	X5			

Табл. 3

б) Псевдослучайно процедурно генериране на пещери

Тези случайно навързани елементи описани горе ще репрезентират пещери. Те се генерират по следния начин:

Започваме с една матрица от елементи (клетки), като всеки елемент има две състояния 1 и 0. Първоначално всички елементи са в състояние 0 (тези които са в състояние 0 ще оставя празни).

След това се генерира на случаен принцип число от 0.0 до 1.0. Взема се хеш кода на това число и на базата на него се генерират случайни числа между 1 и 100. За всеки един елемент от матрицата сравняваме генерираното число със зададен от нас коефицент(между 1 и 100). (Табл. 4)

	1		1			1
1	1		1	1		1
		1				1
1			1	1	1	
1	1			1		
	1		1			1
		1		1		

Табл. 4

Също така имаме опцията вместо да генерираме случайно число от 0.0 до 1.0 да въведем нещо ръчно. Така при еднакъв вход (хеш код) получаваме еднакъв резултат.

В следващата стъпка правим един едностъпков клетъчен процес (на принципа на клетъчен автомат).Обхождайки матрицата засичаме кои са съседите на всяка клетка и каква стойност имат съседите й. Минавайки през елементите поставяме условие от което зависи състоянието на съответната клетка. Вземаме за пример подчертаната клетка от Табл. 6.

0	1	0
0	**1**	1
1	0	0

Табл. 6

Съответно ако сбора от състоянията на съседите на нашата клетка е по-малък от 4, тя променя състоянието си на 0, а ако е по-голям от 4 – го променя на 1. В случай че сбора е точно 4, състоянието й не се променя.(Табл. 6)

Обхождайки матрицата проверяваме дали съответната клетка се намира в периферията й. Ако да – правим състоянието й строго единица. Така правим пещерата затворено пространство, в която нулите са празно пространство, а единиците стени. Резултатът изглежда по подобен начин (с по-голяма матрица).(Табл. 7)

1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1				1	1	1	1	1	1	1
1	1	1								1	1	1	1
1		1	1	1	1							1	1
1				1	1	1	1					1	1
1					1	1	1	1	1	1	1	1	1
1										1	1	1	1
1						1	1						1
1					1	1	1	1					1
1				1	1	1	1	1	1				1
1					1	1	1	1	1				1
1	1	1						1	1				1
1	1	1	1										1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Табл. 7

Елементите със стойност 0 ще представляват пространството в което героя може да се движи, затова трябва да свържем всички отделни области, за да са достъпни.

Първо използваме flood flow алгоритъм за да засечем всяка една група елементи, представляваща област(затворено пространство) .След това запазваме и списък с тези елементи, които лежат в периферията на всяка област.(Табл. 8)

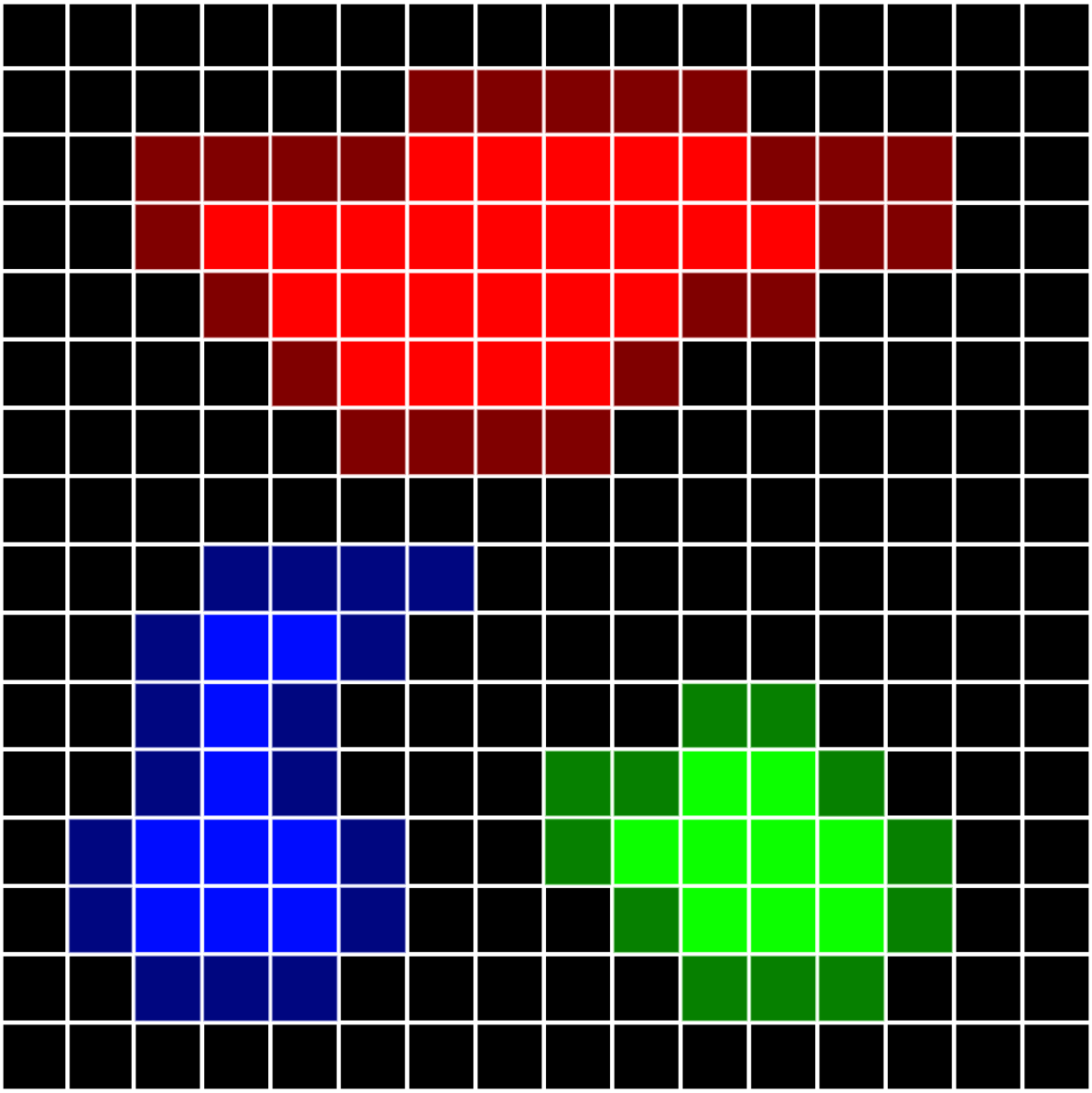


Табл. 8: Всяка област е отбелязана с различен цвят, а периферните й елементи с нюанс на съответния цвят

Сега намираме най близките точки между всеки две области, като сравняваме позициите на периферните им елементи. В случай че има повече от една двойка най–близки точки между две области се избира на случаен принцип едната.(Табл. 9)

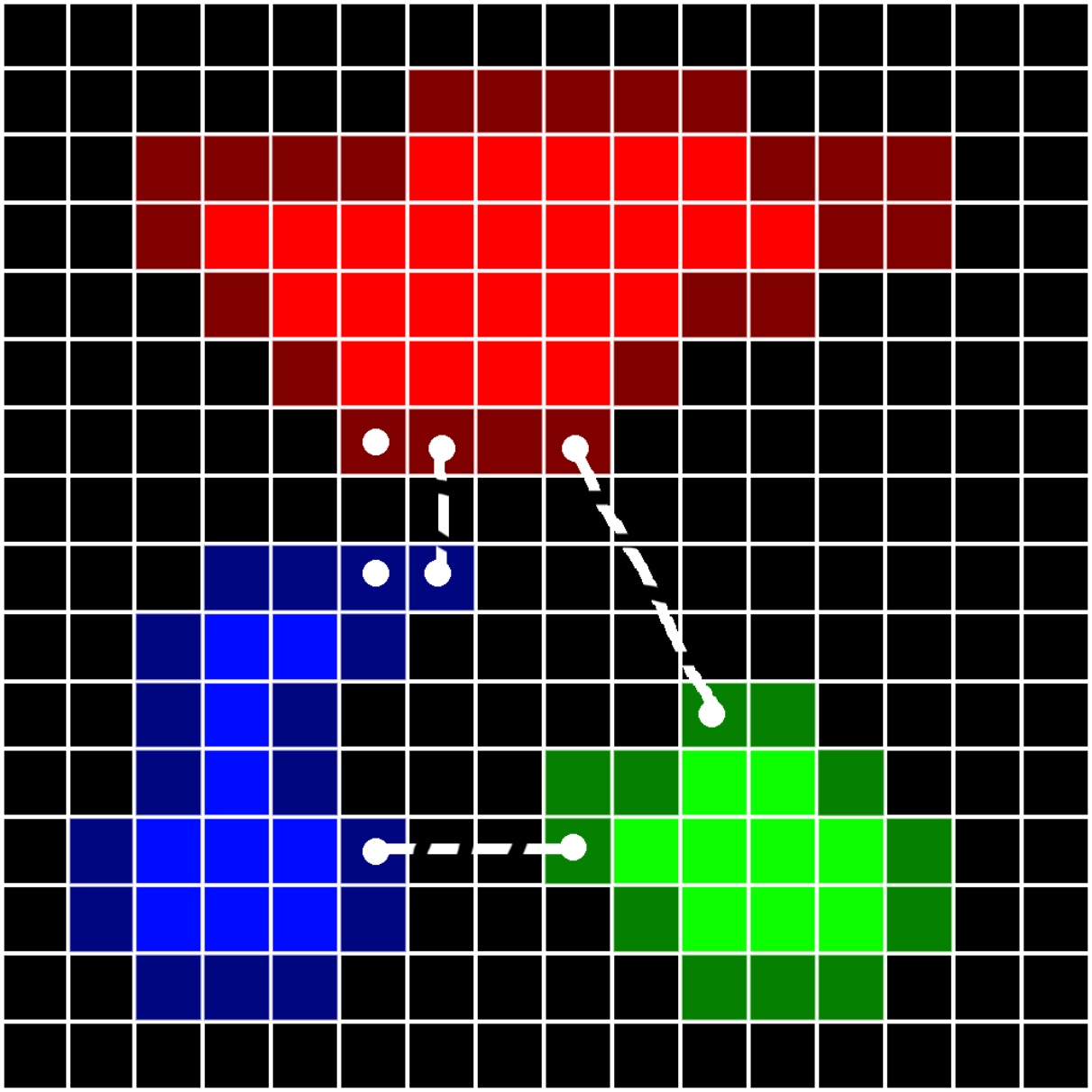


Табл. 9

След като имаме най-късите разстояния, започваме да маркираме коя област с коя ще е свързана. Започваме като свързваме първата(номерирани са според реда в който са засечени при хоризонтално обхождане) с най-близката до нея. Когато стигнем до втората област виждаме че тя вече е свързана със своята най-близка. Затова не правим нищо и преминаваме към следващата.(Табл. 10)

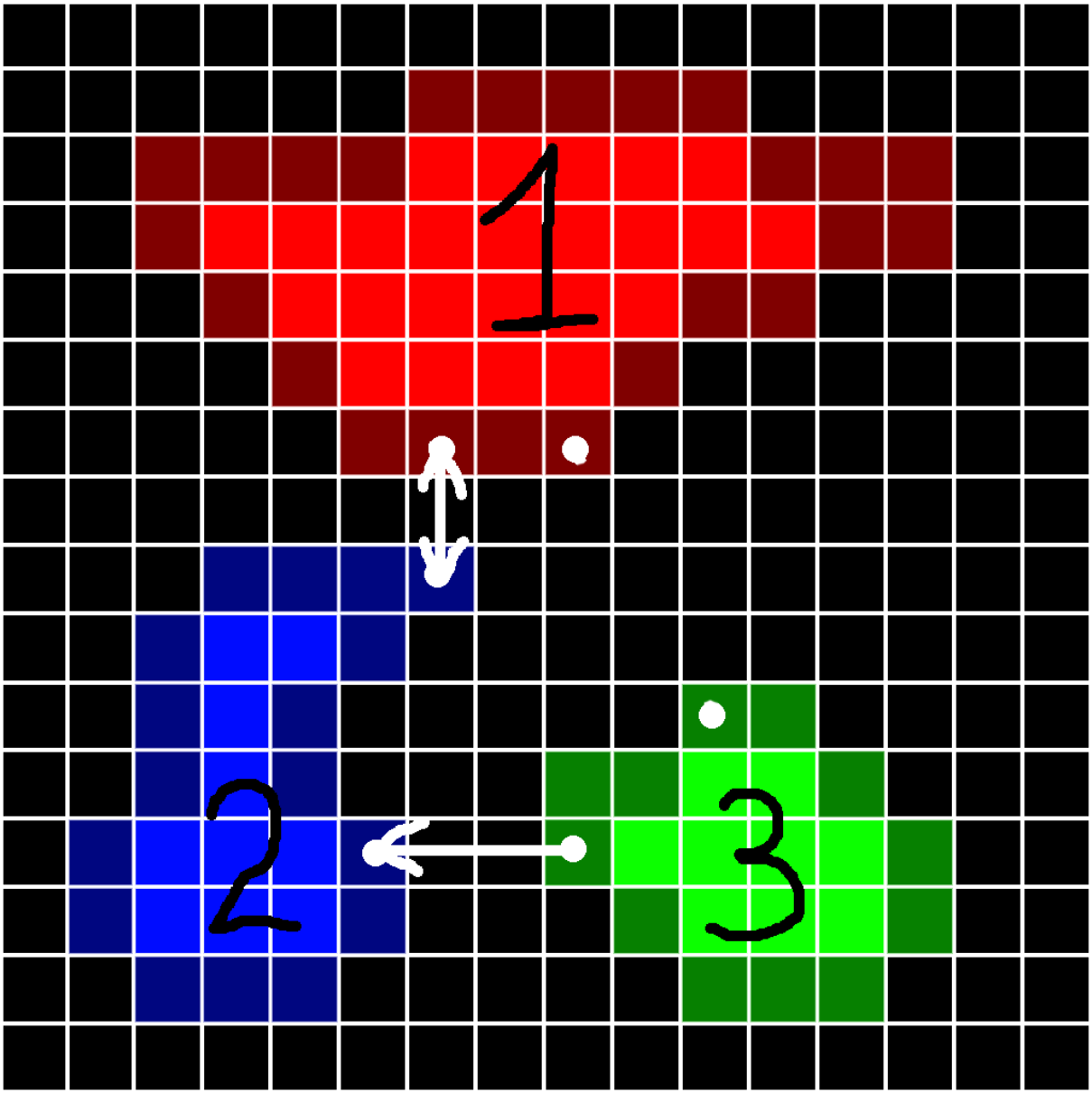


Табл. 10

Сега знаем как трябва да са свързани отделните области, но има ситуация, в която имаме две отделни групи от области.(Табл. 11)

Затова правим втора итерация на свързването, като този път знаем коя област с коя е свързана.

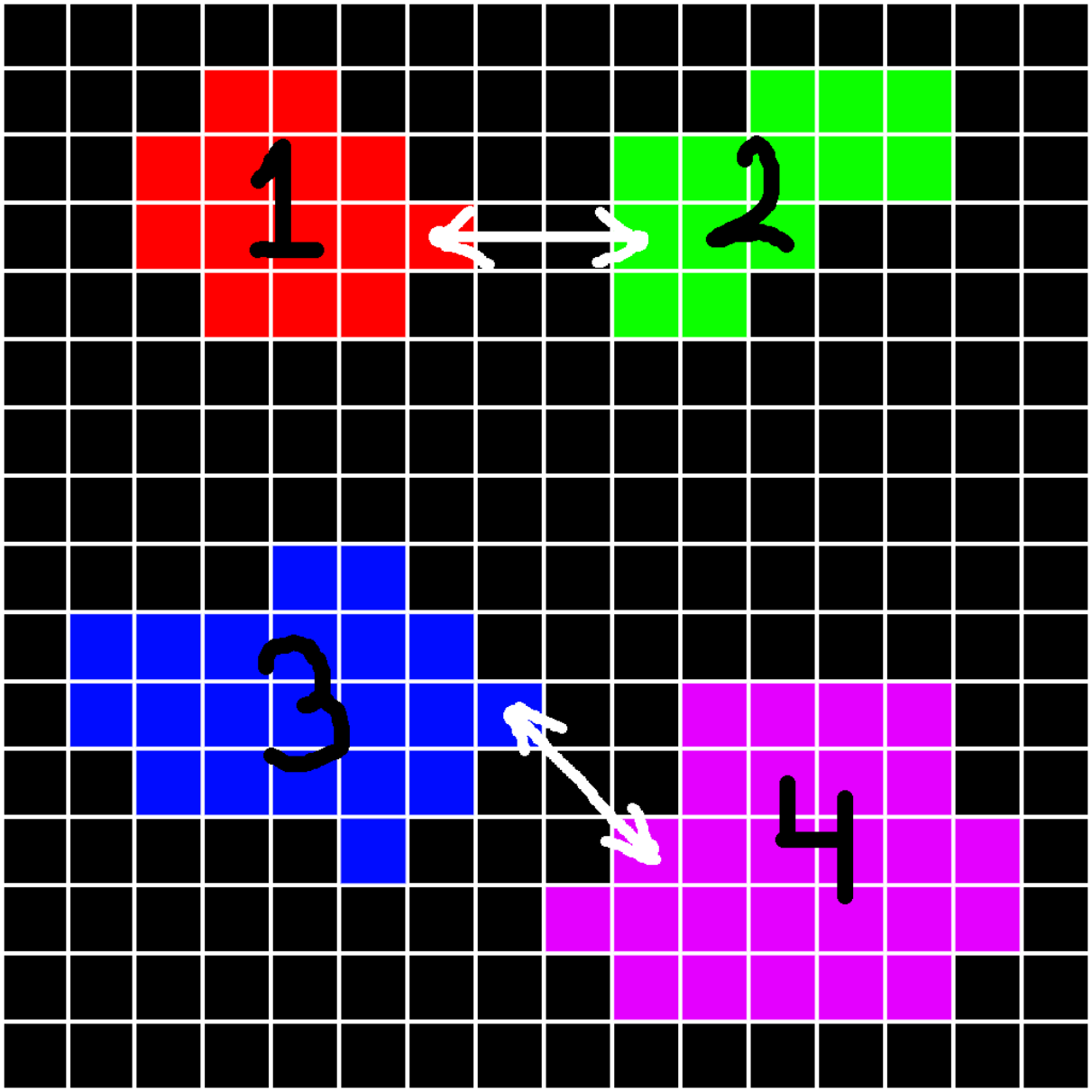


Табл. 11

Тези пътища правим едва след като създадем обектите които да репрезентират тази матрица която имаме. За проправянето на пътища между областите и зползваме Raycast2D, което е лъч или отсечка в равнина, който връща първият обект(съдържащ 2DCollider компонент) в който се блъсне. Създаването на път между две области става като спускам лъчи от едната точка към другата докато следващият спуснат лъч не засече стена и съответно променям ударените от лъча обекти.

в) Алгоритъм за намиране на най-кратък път – A*

A*(А звезда) е евристичен алгоритъм за намиране на най-кратък път между два възела в граф. Тъй като вече разполагаме с матрица(тази на пещерите), ще я използваме като граф за алгоритъма.

Всеки елемент от матрицата може да бъде проходим или непроходим в зависимост от стойността му, определена при генерирането на пещерата. Също така има 3 стойности:

- g(x) – разстоянието от съответния елемент до началния
- h(x) – разстоянието от съответния елемент до крайния
- f(x) – сбора на g(x) и h(x)

Разстоянието между два елемента определяме според това на колко елемента диагонално(стойност от 14 за всеки) и хоризонтално/вертикално(стойност 10 за всеки) се намират.

Алгоритъма започва от началния елемент, като изчислява стойностите на своите съседни елементи.(Табл. 12)

				B						
						14 28 42	10 38 48	14 48 62		
						10 38 48	A	10 52 62		
						14 48 62	10 52 62	14 56 70		

Табл. 12

g(x) – горен ляв ъгъл

h(x) – горен десен ъгъл

f(x) – централно

зелен цвят – налични елементи за избор

Избираме елемента с най-ниска стойност на f(x) и повтаряме действието докато избрания елемент не е равен на крайния(B).(Табл. 13)

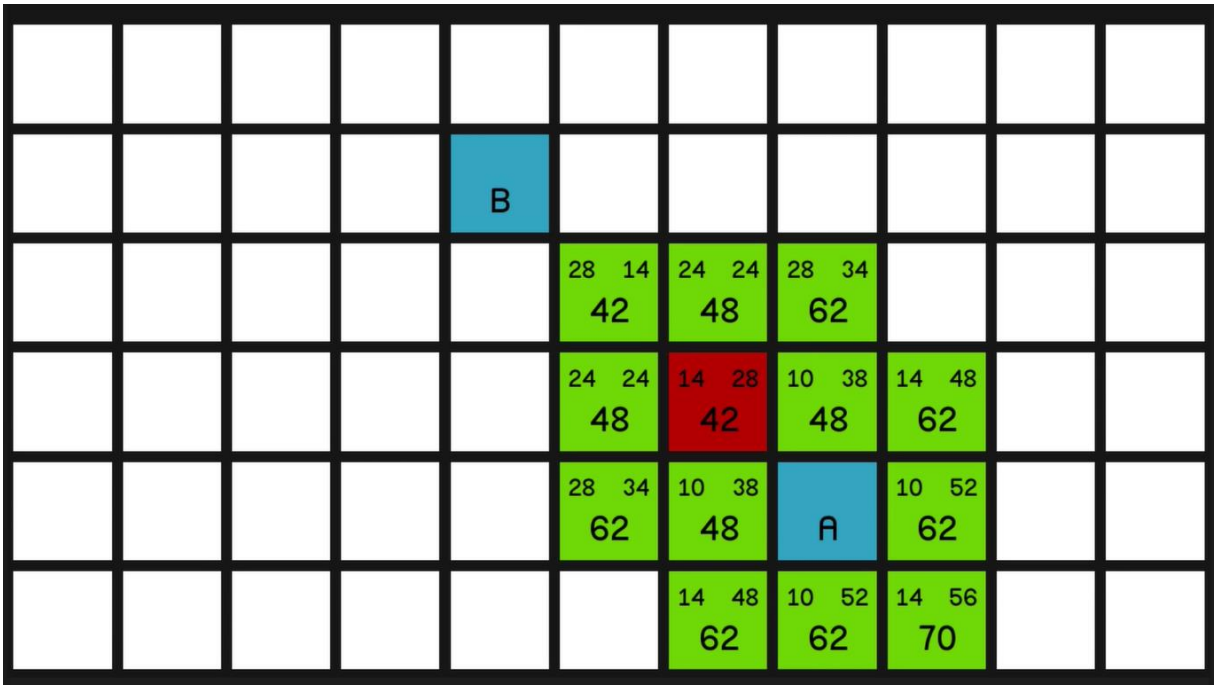


Табл. 13

Червен цвят – избрани елементи

Обаче когато има и непроходими елементи, които трябва да заобиколим можем да има повече от един елемент с най-ниска стойност на f(x). В такъв случай избираме елемента с най-ниска стойност на h(x), но ако тези стойности са равни избираме един елемент на случаен принцип.(Табл. 14)

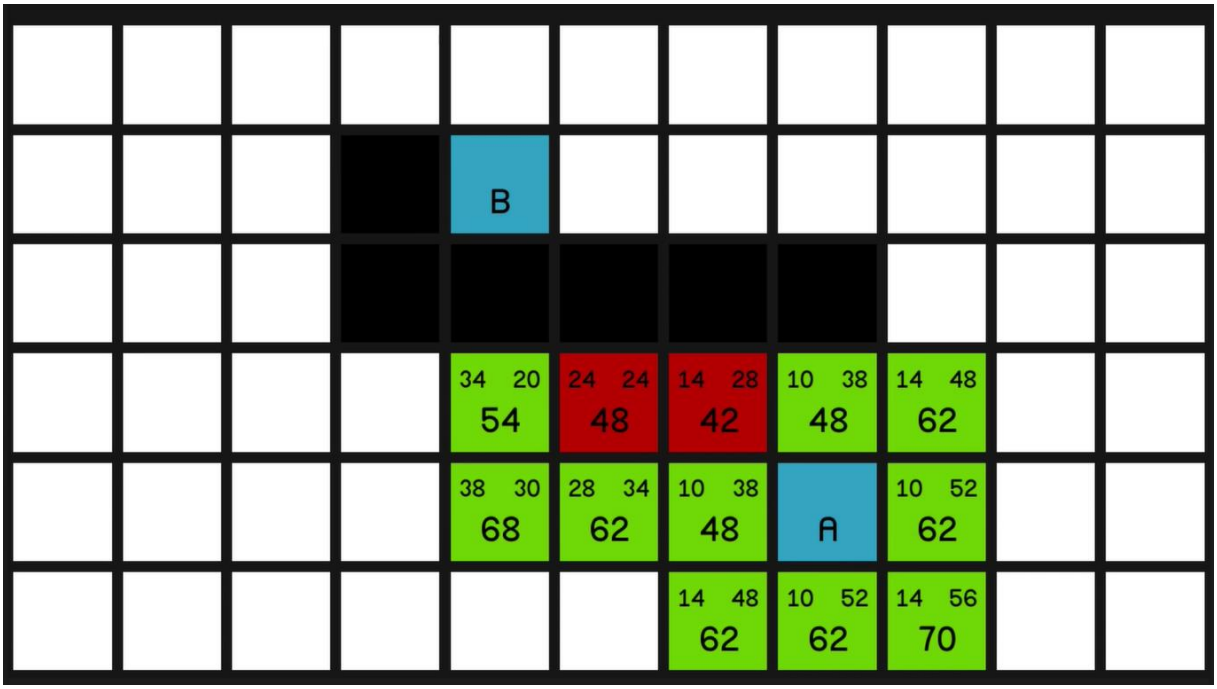


Табл. 14

Преминавайки през елементите, запазваме кой елемент от кой е предхождан. Така когато стигнем до крайния елемент проследяваме пътя до началото и разбираме кои елементи представляват пътя.(Табл. 15)

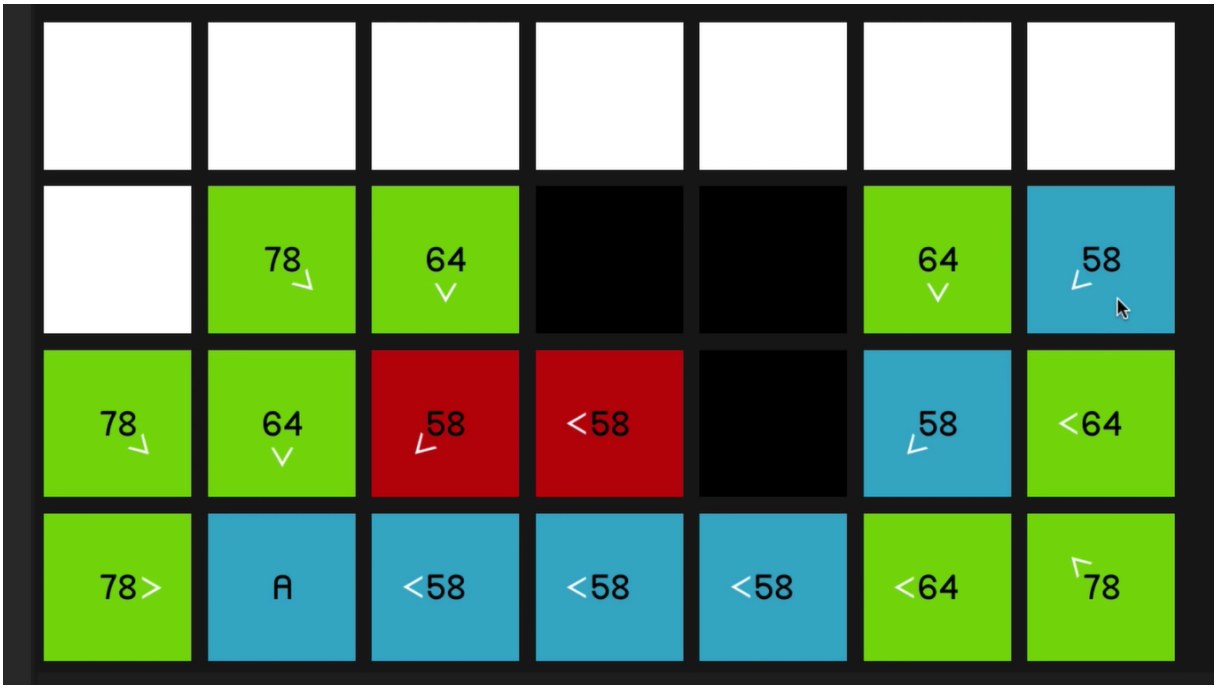
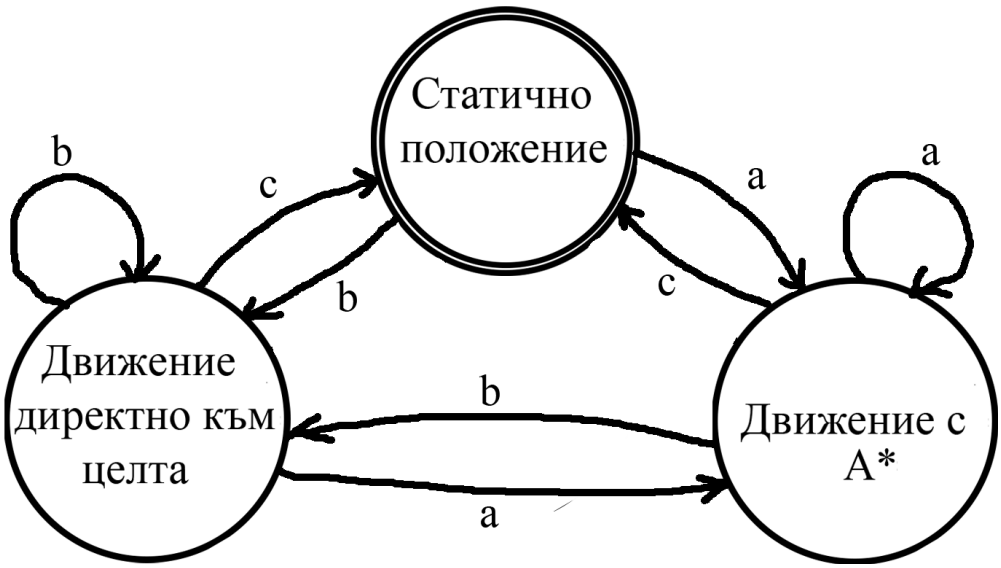


Табл. 15

Син цвят – най-кратък път между A и B

г) Pathfinding

До момента имам едно динамично чудовище(таласъм/гоблин) и съм започнал да работя върху това как той се движи. Той се движи по показаната долу схема.
Условия на автомата:
a – героя е достатъчно близо до чудовището, но то няма пряка видимост към него
b – героя е достатъчно близо до чудовището, но то има пряка видимост към него
c – героя е прекалено далеч от чудовището



д) Health (живот): Health на героя е съставен от контейнери (сърца), които могат да бъдат напълвани или изпразвани (такава концепция се наблюдава при играта The Legend of Zelda). Има 2 вида Health – първият отговаря за живота на нашия герой и ако се изчерпи, героят губи живота си. Вторият има възможността да се регенерира. Освен когато поема damage (щета) от враговете, се изчерпва и когато героят стреля. Ако той е изчерпан, първия вид поема щетата. Начинът, по който работят механиките за Health, е чрез списъци с текущите елементи (контейнери) и техните фази (цяло, половин, празно). Съответните фази се използват като декларирани елементи от масив.

Максимално количество живот и енергия:

7 половинки щета:

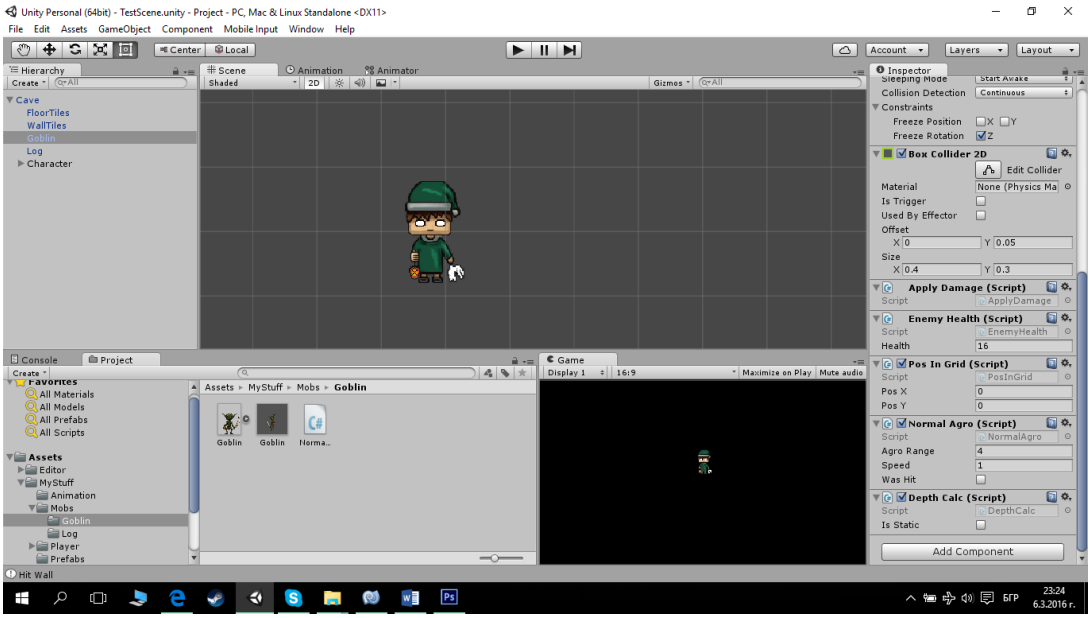
Регенерация на енергията през определен период от време:

- **Разработка**

За разработването на играта съм използвал главно две програми:

- **Unity 3D** е game engine, който е създаден за да бъде достъпен за начинаещи и независими разработчици на игри. Интерфейсът и способностите му са лесни за научаване и същевременно имат достатъчно напреднали функционалности. Още едно от предимствата е, че има функции, специализирани в създаването на 2D продукти. Аз го използвам точно по тези причини и защото е безплатен.

Така изглежда интерфейсът на Unity:



MonoDevelop е вградената среда за програмиране в Unity, предлагаща разширени библиотеки специално за разработка на игри. Дава възможност за писане на езиците - C#, JavaScript и Boo.

- **Adobe Photoshop CS6**

Adobe Photoshop е професионална комерсиална програма за обработка на растерна графика от американската софтуерна компания Adobe Systems. Photoshop позволява интерактивна редакция на сканирани и цифрово заснети графични материали в реално време чрез набор от инструменти. Има собствен файлов формат – .PSD (съкращение на PhotoShop Document), който запазва всички атрибути, използвани по време на работа (например различните видове слоеве (Layers) и маски (Masks) и история на промените (History).

Аз съм използвал Photoshop за създаването на всички графични обекти, използвани в разработката ми. Използвах го, защото е водеща програма за графична обработка и дизайн. Програмата е достъпна за мен, защото училището ми има лиценз.

- **Бъдещо развитие**

В бъдеще ще добавя:

- Пасивни и активни предмети, които ще са бонус или недостатък за играча
- Обекти - камъни, дървета, храсти и др.

- Още врагове и босове (AI вариации)
- Интерфейс
- Crafting система
- Оформен графичен дизайн за всичко

Основни цели в настоящата разработка:

- Доработване на изкуствения интелект
- Дооформяне на цялостната концепция
- Работа върху графичния дизайн

Реализация:

След като Into The Woods бъде завършена, тя ще може да намери своето място в гейминг индустрията чрез процеса **Greenlight** на платформата Steam. Името и интерфесът на играта ще са изцяло на английски, защото Steam е интернационално използван софтуер.

Steam е дигитално разпространителна, мултиплейър и комуникационна платформа разработена от Valve Corporation. Използва се за дистрибуция на игри и свързана медия онлайн, от малки независими разработчици до най-големите софтуерни компании. Като до момента в Steam има над 3500 игри и броя на потребителите надхвърля 100млн души. Платформата обществени функции, автоматични обновления на игрите, облачно запазване на игри и още много функционалности.

Steam Greenlight представлява процес, при който разработчиците качват свой продукт (заедно със снимков и видео материал), за който потребителите на Steam гласуват. При определен минимален брой гласове, продукта се публикува в платформата.

Друга алтернатива, която може да помогне за реализацията на проекта е **Kickstarter** – crowdfunding (практиката да се събират малки суми пари от много хора за

да се финансира проект) платформа. Като не съм правил конкретни планове за стартиране на кампания в Kickstarter.

• Приложение

• Стартиране и системни изисквания

Системни изисквания:

- OS – Windows (XP/Vista/7+)
- DirectX 9+
- CPU: Pentium 4+
- RAM: 512MB+
- Екран с резолюция, по-голяма от 1024 x 768 пиксела

Стартиране:

За да бъде разгледана играта заедно със сорс кода и останалите елементи от разработката е необходимо проекта да бъде отворен с Unity. Изпратеният архив съдържа и инсталационния файл на програмата. За да се стартира проекта трябва да се отвори папката „IntoTheWoods_Project“ с Unity. След това трябва да се отвори сцената:

- Assets/MyStuff/Scenes/TestScene – сцената с герояи двете чудовища в една пещера, която винаги е еднаква (може да промените това от настройките – виж „Сцени от играта“)
- Assets/MyStuff/Scene 0 – сцената с множество пещери (не са налични герой и чудовища)

В папката „MyStuff“ могат да бъдат намерени всичките авторски ресурси.

- **Контроли**

Контроли:

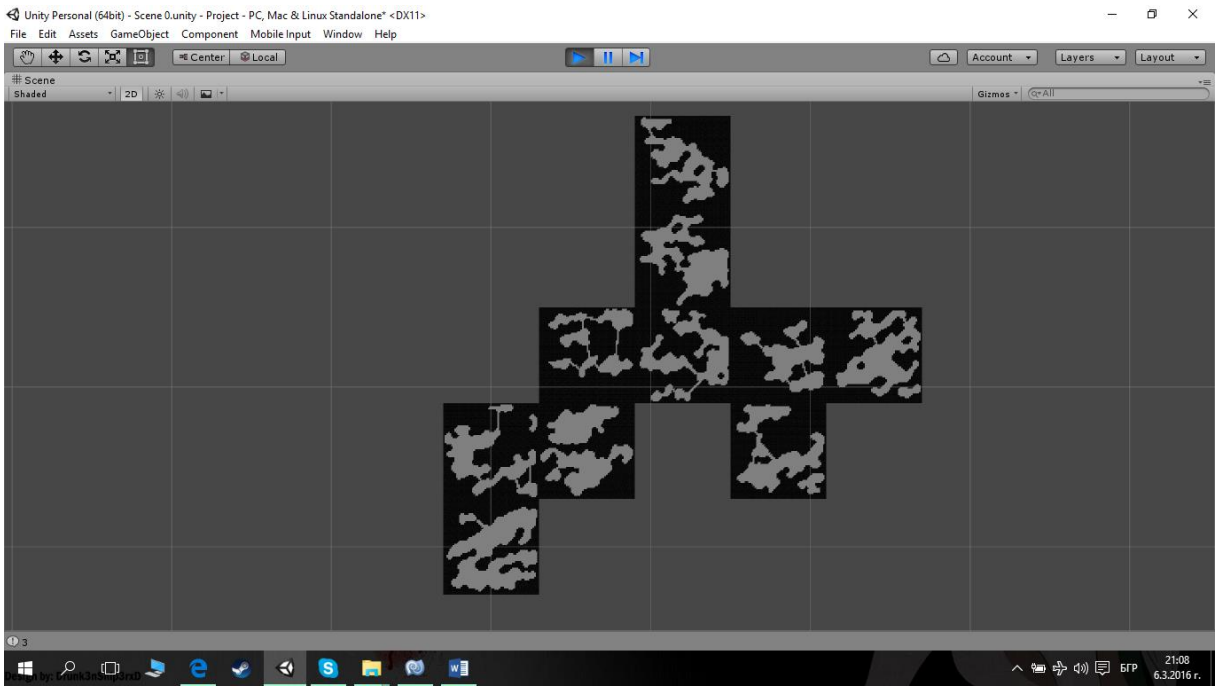
- **W** – движение нагоре
- **S** – движение надолу
- **A** – движение наляво
- **D** – движение надясно
- **↑ –** стрелба нагоре
- **↓ –** стрелба надолу
- **← –** стрелба наляво
- **→ –** стрелба надясно
- **Space –** за преминаване през врата

Тъй като играта все още е в начален стадий на разработка, все още няма цел, която играчът да следва.

- **Сцени от играта**

- От сцената „**Scene 0**“

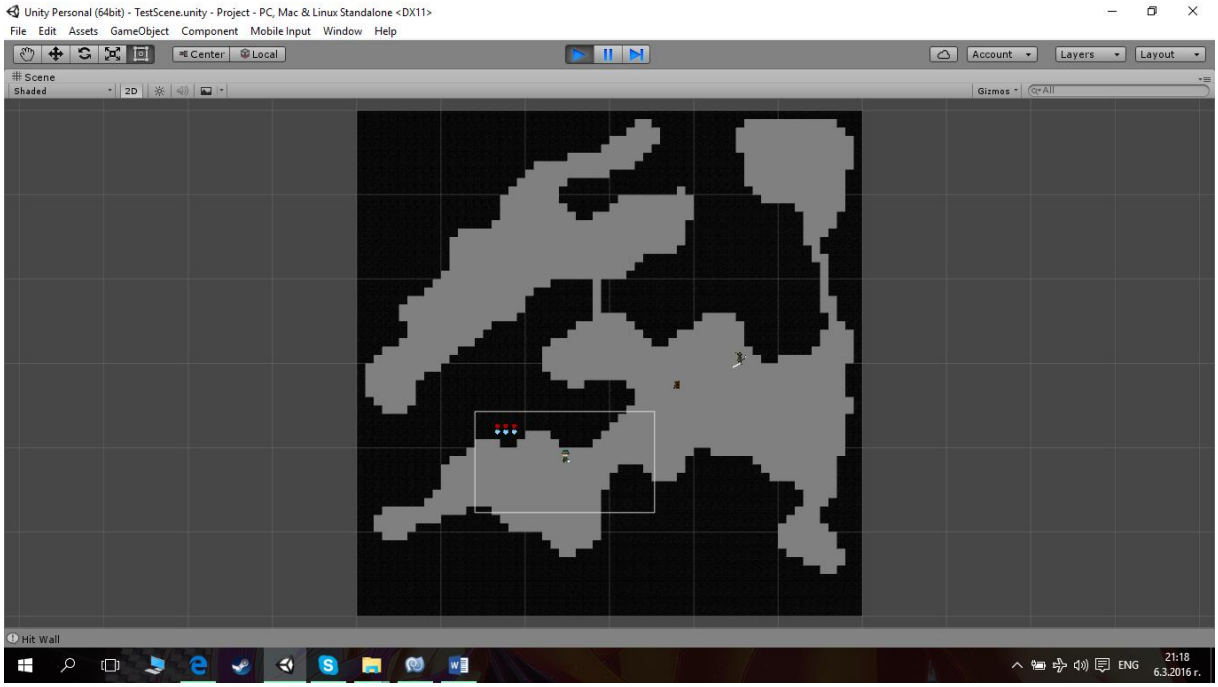
На Фиг. 1 може да видите веригата от елементи(пещери)



Фиг. 1

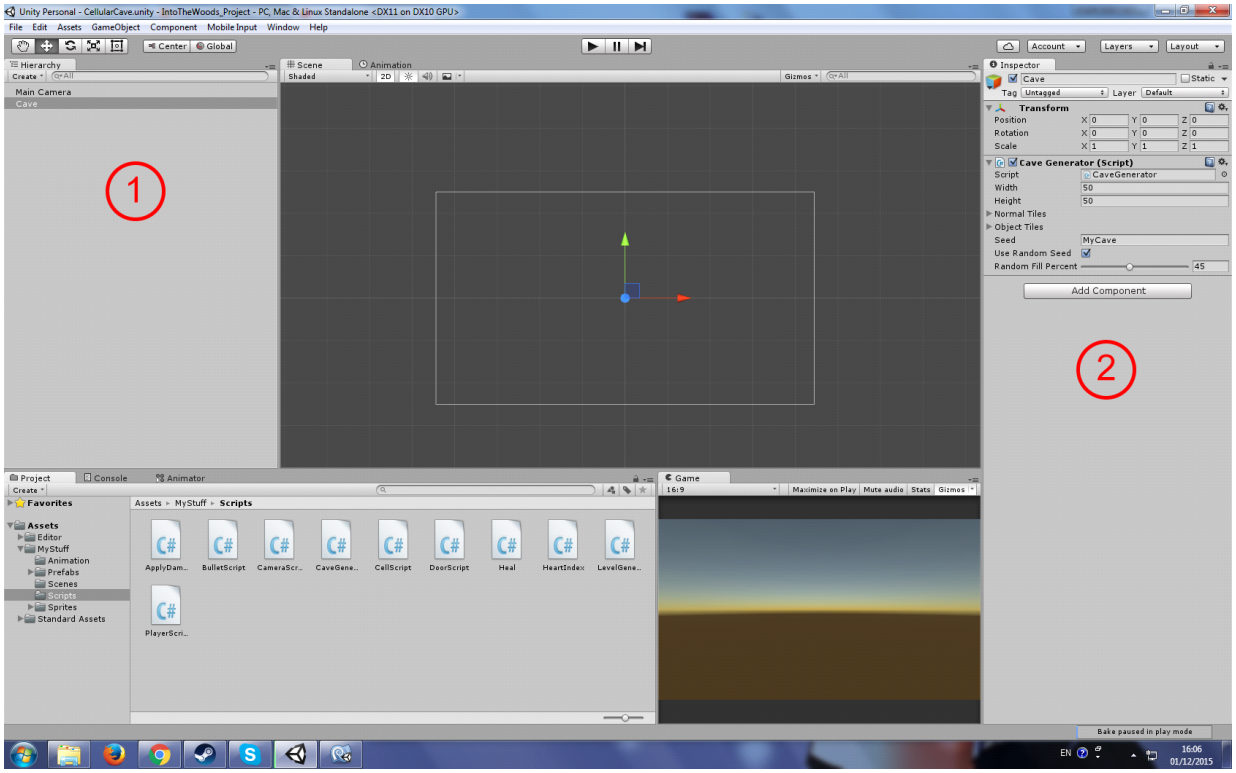
- От сцената „**TestScene**“

На тази сцена се генерира една пещера, героя заедно с оръжието си(лампата, която държи), камерата и сърцата които индикират живота и енергията му. Пещерата винаги е една и съща, но това може да се промени от настройките(има вероятност героя да бъде заклещен в стена). Също така на сцената има един таласъм, който следва героя и му нанася щета при допир и пън, който ущетява енергията на героя когато той се приближи. (Фиг. 2)



Фиг. 2

Допълнителни настройки за генерирането може да намерите като първо изберете надписа „**Cave**“ в полето обозначено с единица (Фиг. 6). След това в полето обозначено с двойка (Фиг. 3) ще видите C# скрипт на име „CaveGenerator“, под който ще има няколко настройки.



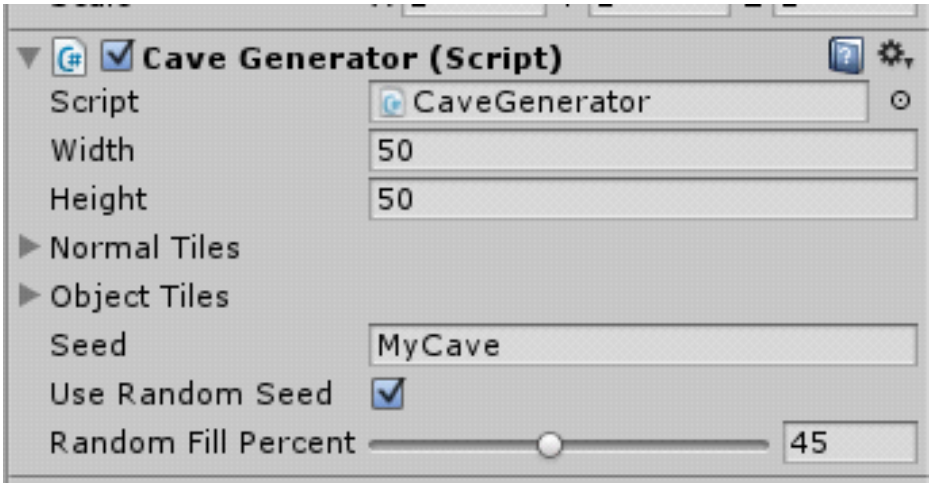
Фиг. 3

Това са следните настройки (Фиг. 4):

- Width, Height – широчина и дължина на матрицата в брой елементи
- Use Random Seed – ако няма отметка пещерата ще се генерира спрямо стойността в полето „Seed“. Ако полето е отметнато, то всеки път ще се генерира спрямо случайна стойност между 0.0 и 1.0
- Seed – вход на стойност спрямо която да се генерира пещерата
- Random Fill Percent – това е приблизителната част от елементите (в проценти), на която ще бъде променено състоянието в първия етап от генерирането (преди прилагането на клетъчния процес)

Забележка:

- Докато правите промени се уверете, че сте спрели играта
- Има много малък шанс не всички области на пещерата да бъдат свързани успешно, но това ще бъде оправено в най-скоро време



Фиг. 4

Когато гоблина използва A* за да намери пътя до героя, елементите които представляват този път са очертани със зелен цвят. Бялата линия, която излиза от него индикира посоката, в която се пуска лъч за да се провери дали има пряка видимост към героя, за да тръгне към него ако има.(Фиг.4)

Фиг. 4

На Фиг. 5 може да видите състоянието на статичното чудовище, когато сте близо до него и когато не сте.

Фиг. 5

- **Използвана литература**

- www.wikipedia.org

• Заключение

Резултатът е една изградена основа, върху която има място за реализация на много идеи. Един продукт, който, ако бъде завършен, ще може да намери своето място на пазара и ще бъде добре оценен от своята целева група. Into The Woods има понтенциала да се отличи с една уникална концепция.