**BUBBLE TROUBLE**

Проектна задача по Визуелно програмирање

Ѓорѓина Цениќ 161517, 2018 година

Github: https://github.com/gjorgjinac/BubbleTrouble

Комплетно самостојна имплементација на игра слична на Bubble Trouble (Bubble struggle) што се состои од играч што погодува топчиња во движење и собира поени сѐ додека не ги елиминира сите топчиња. Во тој случај го поминува нивото. Играчот може неуспешно да ја заврши играта и да загуби живот во еден од овие случаи:

1) Истекување на времето.

2) Играчот е погоден од топките што треба да ги елиминира.

3) Играчот е погоден од бомбите што самиот ги пука.

4) Играчот паѓа од скала или од платформа.

5) Играчот е погоден од тенк

6) Играчот е погоден од паѓачкиот ѕид

Играта завршува кога играчот ќе ги изгуби сите животи.

Поставени се default контроли за раководење со играчот, а во менито има и опција за менување на истите.

|  |  |
| --- | --- |
| Left arrow key | Движење на играчот лево |
| Right arrow key | Движење на играчот десно |
| Up arrow key | Движење на играчот горе (опција само при движење по скала) |
| Down arrow key | Движење на играчот долу (опција само при движење по скала) |
| Space | Пукање |

При секој погодок, поголемите топки се разделуваат на 2 помали, сѐ додека радиусот им е поголем од 20. При секое погодување на топка, се добиваат 50 поени.

Дополнително, играта вклучува и ‘’goodies’’ што паѓаат на рандом интервали, се задржуваат на земјата по 3 секунди и потоа исчезнуваат. Ваквите ‘’goodies’’ вклучуваат:

|  |  |
| --- | --- |
|  | coins-плус 30 поени. |
|  | cash-плус 100 поени |
|  | pizza-плус 1 живот |
|  | shield-10 секундна заштита од топчиња |
|  | Time- плус 10 секунди време |

На финалните поени на играч што победил некое ниво на играта се додаваат плус 10 поени за секоја неискористена секунда од времето.

На играчот му се нуди опција да го сочува својот резултат (и при успешно и при неуспешно завршување на ниво) во листата со High Scores што се серијализира при секое ново додавање на играч.

Менито обезбедува функционалности за отварање на прирачникот за играње, паузирање, затворање, сочувување и отварање на веќе сочувана игра.

Копчето Settings овозможува бирање на тежината на играта, изгледот на играчот (во зависност од играчот се менува и позадината) и менување на контролите за движење и пукање. Во зависност од избраната тежина се менуваат бројот, големината и брзината на движење на топчињата.

Ниво 1:

Целта на играчот е да ги елиминира сите топчиња, со тоа што мора да се искачи по скалата за да го елиминира ,,специјалното’’ топче што се одбива од препреката. Бомбите исто така се одбиваат од препреката и не можат да стигнат до специјалното топче освен ако играчот не се качи по скалата. Играчот исто така губи живот ако падне по скалата, падне од десниот раб на препреката или биде погоден од тенкот или од бомбите што се одбиваат од препреката.

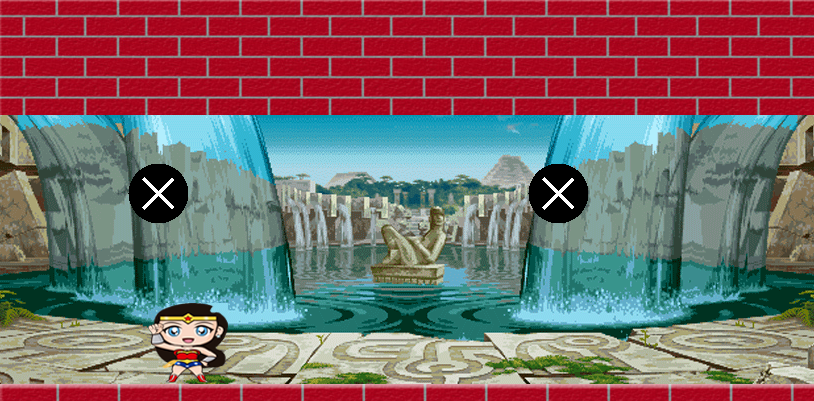


НИВО 2:

Движењето на топките и на играчот се ограничени со препреки, со тоа што играчот може да премине низ секоја препрека само откако ќе го исчисти претходната област со топчиња.

НИВО 3:

На ова ниво предизвикот е паѓачкиот ѕид од кој се одбиваат и топчињата, и бомбите што ги пука играчот. Така што, ако играчот не е прецизен при пукањето, бомбата може да му се одбие од ѕидот и да го убие.



Имплементација:

Класите Obstacle, Bullet и Bomb наследуваат од класата Circle и ги имаат основните компоненти: радиус, боја, насока на движење и позиција. Класата Obstacle се однесува на топчињата што треба да бидат уништени, Bullet се куршумите што ги пука тенкот, а Bomb се бомбите што ги пука играчот. Дополнително, класата Obstacle содржи и родител (типот на топче по чие распаѓање е добиена) и вредности за хоризонтална и вертикална насока. Класата Bomb дополнително содржи и почетна позиција што е потребна во првото ниво за одредување кои бомби ќе се одбиваат од препреката.

Трите класи обезбедуваат функционалности за движење, исцртување, проверка за колизија со играчот или со други топчиња.

Класата Game содржи податоци за изминатото време, бројот на преостанати животи, листа од тековно активните бомви, бројот на освоени поени, името на играчот, тежината, нивото и објект од класата Obstacles (помошна класа што чува листа од Obstacle и ја олеснува нивната манипулација).

Класата Minion се користи за чување на сликите за играчите во различна ситуација.

Класата HighScore ги зачувува сите постигнати резултати и ги покажува најдобрите 10. За тоа користи серијализација и при секое отварање на формата резултатите се читаат од фајл. За додавање на нов играч се користат класата Player (што го чува името и поените на играчот) и формата AddPlayer (што го зема како инпут името на играчот). Поените се препраќаат од главната форма кон формата AddPlayer за на крај да се сочуваат во класата Player и листата со резултати.

Објаснување на функцијата за движење на човечето што се извршува при настанот KeyDown: (Објаснувањето го има и во самиот проект)

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

Ako na igracot mu e dozvoleno da go pomrdne coveceto (pri paganje od skala ili od platforma mora da poceka da stigne do zemjata, ne moze da "leta",togas controlLock se postavuva na true i se zaklucuvaat kontrolite)

if (!controlLock)

{

Ako pristisnatoto kopce e kopceto nameneto za pukanje, se dodava bomba

if (e.KeyCode == shoot)

game.AddBomb(player.Location);

Postojat razlicini implementacii za razlicni nivoa bidejki ima razlicni barieri

if (level == 1)

{

Pri dvizenje levo ili desno na prvo nivo, samo treba da se onevozmozi izleguvanjeto na igracot od prozorecot

if (e.KeyCode == left && player.Location.X > 0)

player.Location = new Point(player.Location.X - 10, player.Location.Y);

if (e.KeyCode == right && player.Location.X < Width - 80)

player.Location = new Point(player.Location.X + 10, player.Location.Y);

Igracot e pictureBox so visina 80. Za dvizenje nagore se proveruva dali igracot se naogja pod ili na skalata (razlikata megju X koordinatite da bide pomala od 100). Vtoriot del od uslovot (player.Location.Y + 70 > platform.Location.Y) go sprecuva igracot da se iskaci nad nivoto na platformata, da ne lebdi

if (player.Location.X - ladder.Location.X < 100 && player.Location.Y + 70 > platform.Location.Y && e.KeyCode==up)

player.Location = new Point(player.Location.X,player.Location.Y-10);

Za dvizenje nadolu isto taka se proveruva dali igracot se naogja na skalata. Vtoriot del od uslovot (player.Location.Y + 70 < Height - 100) go sprecuva igracot da se spusti premnogu nisko vo prozorecot

if (player.Location.X - ladder.Location.X < 100 &&

player.Location.Y + 70 < Height - 100 && e.KeyCode == down)

player.Location = new Point(player.Location.X, player.Location.Y+10);

Sledniot uslov go ovozmozuva pagjaneto na igracot ako toj pri simnuvanjeto izleze od granicite na skalata ili pri dvizenjeto desno na platformata izleze od nejziniot rab

Coveceto se naogja nad platformata i negovata X koordianta e pogolema od sirinata na platformata

if ((player.Location.Y < platform.Location.Y && player.Location.X > platform.Width)

Coveceto se naogja pod platformata, a nad zemjata, a X koordinatata mu e pogolema od krajnata tocka na skalata (Izlezeno e od rabot na skalata)

|| (player.Location.Y > platform.Location.Y && player.Location.Y < Height - 175 && player.Location.X > ladder.Location.X + ladder.Width - 30))

{

if (e.KeyCode == right)

{

Se zaklucuvaat kontrolite se dodeka coveceto ne padne na zemjata

controlLock = true;

Se menuva izgledot na coveceto so otvorena usta kako da vreska

player.Image = minions[selectedPlayer].screaming;

}

}

}

Vtoro nivo

if (level == 2)

{

Na vtoroto nivo ima samo dvizenja na levo i desno, poradi toa sto nema skali

Za dvizenja na levata strana nema ogranicuvanja poradi toa sto barierite se naogjaat na desnata strana i istite ne se zatvoraat otkako ednas ke se otvorat. Edinstveno samo se onevozmozuva coveceto da izleze od leviot rab na prozorecot

Igracot samo se pridvizuva za 10 tocki na levo

if (e.KeyCode == left && player.Location.X > 0)

player.Location = new Point(player.Location.X-10,player.Location.Y);

Za dvizenja na desnata strana treba da se razgleda sostojbata na preprekite

if (e.KeyCode == right && player.Location.X < Width - 80)

{

Voveduvame boolean promenliva sto ke odluci dali coveceto moze da se pomrdne nadesno

Ja inicijalizirame na false bidejki ima pomalku situacii koga coveceto moze da se dvizi

Boolean canMove = false;

Ako coveceto se naogja pred prvata prepreka, ne ne interesira sostojbata na preprekite

I coveceto moze da se pomrdne na desno

if (player.Location.X < barrier1\_1.Location.X - barrier1\_1.Width - 40) canMove = true;

Ako dvete prepreki se otvoreni, coveceto moze da se pomrdne

if ((!barrier1\_1.Visible && !barrier2\_1.Visible)) canMove = true;

Ako prvata prepreka e otvorena, a coveceto se naogja pomegju prvata i vtorata prepreka ili pred prvata prepreka, moze da se pomrdne

barrier2\_1.Location.X - barrier2\_1.Width - 40 e pozicijata na vtorata prepreka (-40 poradi radiusot na coveceto)

if ((player.Location.X < barrier2\_1.Location.X - barrier2\_1.Width - 40 && !barrier1\_1.Visible))

canMove = true;

Coveceto se pomrdnuva na desno

if (canMove)

player.Location = new Point(player.Location.X + 10, player.Location.Y);

}

}

if (level == 3)

{

Pri dvizenje levo ili desno na treto nivo, samo treba da se onevozmozi izleguvanjeto na igracot od prozorecot

if (e.KeyCode == left && player.Location.X > 0)

player.Location = new Point(player.Location.X-10,player.Location.Y);

if (e.KeyCode == right && player.Location.X < Width - 80)

player.Location = new Point(player.Location.X+10, player.Location.Y);

}

}

Invalidate(true);

}