TETRIS

Изработено од Александар Сандев и Андреј Генчовски

Опис на проблемот и упатство за играње:

Апликацијата која што ја развиваме е играта „Тетрис“ во која има неколку модови на игра и тоа:

* Еден играч – Single Player
* Двајца играчи – Two Players
* Екстремен левел – Extreme Mode

A blue rectangular sign with black text

Description automatically generated

Играта се игра на начин, што на одредено време се генерира фигура најгоре на игралиштето, и таа паѓа надолу сама, или корисникот може да ја симне побргу. Има можност за движење на фигурата лево или десно, како и ротирање на истата. Кај еден играч и екстремен левел, командите се следни:

* На горната стрелка се ротира за 90 степени
* На долната стрелка се поместува фигурата надолу
* На левата стрелка се поместува налево за една позиција
* На десната стрелка се поместува надесно за една позиција

Кај двајца играчи, контролите за вториот играч се истите, но копчињата се “w”, ”s”, “a”, “d” соодветно. Контролите се прикажани на екран при секое вклучување на соодветниот мод на игра, па така корисникот нема да има дилема околу тоа. Кога еден ред се исполни со фигури, редот се брише и се додаваат соодветни вредности на атрибутите за поени, ниво и редови. Доколку над тој ред има други фигури, тогаш тие се поместуваат за едно место надолу.

На почетокот, корисникот има копче за помош каде добива соодветни информации.

A blue sign with black text

Description automatically generated

При старт на играта во првите два начини на игра, на почетокот корисникот избира тежина на игра. На играчот му се понудени три нивоа на тежина и тоа:

* Лесно - Easy
* Средно - Intermediate
* Тешко – Hard

A screenshot of a computer

Description automatically generated

Разликата помеѓу секое од овие нивоа е брзината на генерирање на фигура и движењето надолу. Корисникот исто така има прозорче каде има преглед за следната фигура која што ќе се генрира.

A colorful squares on a blue background

Description automatically generated

Првите 4 редови се обоени во црвена боја поради тоа што кога веќе фигурата се наоѓа таму и не може да оди надолу, тогаш играта завршува. Полето за игра е јасно обележано и боите не се преклопуваат. Корисникот има можност да види каде ќе падне фигурата, преку обележани црвени линии на полињата. За секој играч (еден или два) се прикажуваат податоци за име, редици, поени и ниво. Овие вредности имаат улога во одредувањето на победникот кај двајца играчи или освоени поени кај еден играч.

A screenshot of a computer

Description automatically generated

За време на играта, на корисникот му е прикажан модот на тежина на кој игра.



При крај на играта, по соопштувањето на поените или победникот, корисникот добива прашање дали сака да игра на истиот мод на игра повторно.

A screenshot of a computer

Description automatically generated

Со потврда на прашањето, играта почнува од ново, а со одбивање се враќа на почетното мени. Копчето за излез го прашува горисникот дали е сигурен дека сака излез, и со потврда играта се исклучува.

A screenshot of a computer

Description automatically generated

Освен тоа, корисникот има можност да избере на кој јазик сака да ја игра играта, поточно може да избере македонски или англиски. На почетокот е поставен англиски јазик.

A screenshot of a computer

Description automatically generated

Кај есктремниот левел, нема приказ на следните фигури, нема полиња во црвена боја на почетокот, нема приказ каде ќе падне фигурата и брзината на генерирање на фигура и придвижувањето надолу е побрзо од вообичаено. Притоа, колку повеќе играчот ја игра играта, брзината се зголемува се повеќе и повеќе.

Опис на решението на проблемот и Опис на функции и класи од изворниот код:

Нашата игра функционира така што на врвот на игралиште се појавува фигура која играчот може да ја движи, односно да ја поместува лево, десно, надолу или да ја ротира. Оваа фигура паѓа надолу на определен интервал и кога ќе го допре дното на игралиштето или некоја друга веќе падната фигура, таа застанува и се смета за падната, истовремено се проверува дали го допира врвот на игралиштето, ако го допира врвот, играта завршува, ако не го допира врвот, се креира нова фигура и играта продолжува.

За имплементација на фигурата користеме апстрактна класа Shape од која наследуваат класите Shape4, ShapeL, ShapeLine, ShapeSquare и ShapeT. Со овие класи овозможуваме креирање на формата, определување бојата и движење и ротирање на фигурата.

Во апстракната класа Shape, за креирање на формата чуваме матрица boolean вредности со големина 4x4, преку која е претставена формата на фигурата. Покрај матрицата чуваме податоци за почетен хоризонтал и вертикален индекс за матрицата boolean вредности, како и за должината и ширината на фигурата. Овие податоци се пополнуваат од страна на наследниците класи во методot public abstract void FillMatrix(). За движење и ротирање на фигурата се чуваат дополнителни информации, а тоа се индексот на редицата и колоната на фигурата во игралиштето, исто така се чуваат лимитите за колку може да се движи фигурата лево и десно. За ротација чуваме променлива Stage која ни кажува како е ротирана формата. Класата Shape исто така го имплементира методот public abstract object Clone().

Движењето на фигурата лево, десно или надолу се прави со методите public void MoveLeft(),public void MoveRight() и public void MoveDown(), така што се намалува или зголемува индексот на колоната или редот во игралиштето, внимавајќи на лимитите за лево и десно со помош на методот public bool Limits(). При движење надолу, ако фигурата дошла на дното, променливата public bool AtBottom { get; set; } се поставува на true. Ротирањето се извршува во методот public bool Rotate(), во овој метод се проверуваат лимитите и ако успешно се ротирала формата се враќа true, а ако неуспешно се ротирала се враќа false.

За имплментација на игралиштето ги користеме класите GridSquare и Playground. Со овие класи го правиме игралиштето и просторот за приказ на следната фигура. Тие се претставени како матрици од коцки. Овие две класи ги содржат методите потребни за приказ на игралиштето и на фигурите.

Класата GridSqaure е помошна класа која ја претставува коцката во матриците коцки, преку кои се претставени игралиштето и просторот за приказ на следната фигура. Во оваа класа се чуваат податоци за приказ на секоја коцка, а и податоци дали коцката е дел од фигура или не со помош на променливата public bool IsPartOfShape { get; set; } и дали коцката е дел од приказот за каде ќе падне фигурата со променливата public bool IsPartOfBottomPreview { get; set; }.

Покрај правење на игралиштето, класата Playground има многу други функционалности, во неа се генерира нова фигура, се проверува дали фигурата е на дно, се проверува дали играта е завршена, се поместува фигурата низ игралиштето, се прави приказ каде би паднала фигурата и некои други функционалности потребни за да може играта да функционира.

Класата Playground ги содржи најважните функционалности на играта. Во неа преку матрици коцки се претставени игралиштето - public GridSquare [,] GridMatrix { get; set; } и просторот за приказ на следната фигура - public GridSquare[,] PreviewShapeSquares { get; set; }. Формата која корисникот може да ја движи е претставена со променливата public Shape MovingShape { get; set; }.

Во класата Playground генерирање на нова фигура се прави преку методот public Shape GenerateShape(GridSquare dot, int index). Во овој метод ни е потребно да користиме Random па затоа во класата имаме public static Random Random { get; set; } = new Random();. Со овој Random сличајно одбираме број помеѓу 0 и 4, и со помош на switch case, според добиениот број генерираме случајна фигура.

Секогаш пред фигурата MovingShape да се движи надолу, преба да се провери дали е на дно, ако е на дно треба да се постави MovingShape.AtBottom на true. За оваа проверка го користиме методот public void CheckIfMovingShapeAtBottom() кој е специјално направен за MovingShape, тој проверува и ја поставува вредноста на MovingShape.AtBottom.

Ако MovingShape.AtBottom = true тогаш треба да се провери дали играта е завршена, односно дали некоја коцка од MovingShape е на врвот на игралиштето. Ова го проверуваме со методот private void CheckIfGameOver() и резлутатот го сместуваме во променливата public bool GameOver { get; set; }.

За движење на MovingShape се користи методот public void Move(Keys keys), кој прима вредност на претиснато копче од формата TetrisGame, и во зависност од кое копче е претиснато ги повикува соодветните методи за движење или ротирање на MovingShape. Овој метод зависи од тоа дали модот е на еден играч или двајца.

За приказ каде би паднала фигурата го користиме методот private void BottomPreview(). Во овој метод користиме clone на MovingShape - Shape tempShape = (Shape)MovingShape.Clone(), и овој tempShape го спуштаме надоли и проверуваме дали е на дно, ако е на дно таму го правиме приказот за каде би паднала формата MovingShape.

Во оваа класа Playground имаме и други клучни методи за функционирање на играта, тоа се методите:

* public void DrawSquares(Graphics g) – Служи за прикажување на сите елементи на играта
* public void Tick() – Со овој метод се поместува MovingShape надолу или се додава нова фигура
* private void CheckFullRows() – Проверува дали имаме полни редици, ако има ги бриши и ни пресметува добиени поени

Играта се прикажува на формата TetrisGame, а помош менито се прикажува со формата HelpStart. Во формата TetrisGame имаме имплементирано неколку функциналности од играта, од кои главни се определувањето на интервалот на тајмерот, од кој зависи брзината на паѓање на фигурата, според одраната тежина. За одбирање на ниво на тежина се користи дополнителна форма наречена Difficulty.