# Практически проект: Бързо рисуване

.

A person in a uniform

Description automatically generated with low confidence

Бързо рисуване е игра, която тества **времето на реакция**. След като изберете **трудност**, ще се сблъскате с вашия **опонент** (компютъра). Трябва да изчакате да **получите сигнал**, след което да **стреляте** с пистолета си **възможно най-бързо**.

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

## Създаване на GitHub репо

Вече имате създаден **GitHub** акаунт, така че може да преминете директно към **създаването на ново репо** от <https://github.com/new>. Изберете **смислено име**, например "QuickDrawBy{вашето потребителско име}", добавете **кратко описание** и направете репото **публично**. **Добавете** README.md **файл** и .gitignore за **Visual Studio**. **Променете лиценза** на "MIT" и кликнете бутона[Create] за да **създадете репото**.

|  |  |
| --- | --- |
| Icon  Description automatically generated | Моля изберете **оригинално име** за вашия проект!  Вашият GitHub профил трябва да бъде **уникален**, а не същият като тези на съучениците ви.  Можете да следвате този документ, но е позволено да **правите промени** и да **добавяте свои функционалности**. |

Сега вашето **репо е създадено** и трябва да изглежда по подобен начин:

Graphical user interface, text, email, website

Description automatically generated

## Код за играта

### Създаване на проект във Visual Studio

Първо, трябва да **стартираме Visual Studio** и да **създадем нов C# console application**. След това трябва да **изберем подходящо име** и **място за съхранение** на проекта. На следващия екран изберете [.NET 6 (Long-term support)] и създайте проекта.

Graphical user interface, text, application, email

Description automatically generated

Преди да продължим, нека да променим **името** на главния клас - Program.cs – на нещо **по-смислено**. Вече знаете как да го направите:

Text

Description automatically generated

### Имплементиране на логиката на играта

#### Създаване на изглед с анимации

Създайте променлива от тип Random, която ще използваме по-късно, заедно с **няколко константи**, които ще ни помогнат да **нарисуваме** различните изображения на **конзолата**.

Можете да копирате **константите** оттук:

|  |
| --- |
| const string Menu = @"  Quick Draw  Face your opponent and wait for the signal. Once the  signal is given, shoot your opponent by pressing [space]  before they shoot you. It's all about your reaction time.  Choose Your Opponent:  [1] Easy....1000 milliseconds  [2] Medium...500 milliseconds  [3] Hard.....250 milliseconds  [4] Harder...125 milliseconds";  const string Wait = @"    \_O O\_  |/|\_ wait \_|\|  /\ /\  / | | \  ------------------------------------------------------";  const string Fire = @"    \*\*\*\*\*\*\*\*  \* FIRE \*  \_O \*\*\*\*\*\*\*\* O\_  |/|\_ \_|\|  /\ spacebar /\  / | | \  ------------------------------------------------------";  const string LoseTooSlow = @"  > ╗\_\_O  // Too Slow / \  O/\_\_/\ You Lose /\  \ | \  ------------------------------------------------------";  const string LoseTooFast = @"    Too Fast > ╗\_\_O  // You Missed / \  O/\_\_/\ You Lose /\  \ | \  ------------------------------------------------------";  const string Win = @"    O\_\_╔ <  / \ \\  /\ You Win /\\_\_\O  / | /  ------------------------------------------------------"; |

Вече създадените константи с изображенията трябва да изглеждат така:



Text

Description automatically generated

A picture containing table

Description automatically generated

A picture containing diagram

Description automatically generated

A picture containing diagram

Description automatically generated

#### Визуализация на меню за трудност

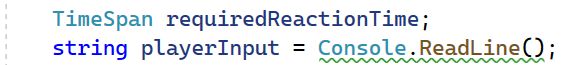
Сега напишете while цикъл. Първото нещо, което ще направим в него, е да използваме **метода** Console.Clear(), който ще изчисти **конзолата**. Също трябва да **отпечатаме** на конзолата вече създадената **константа** Menu.

Graphical user interface, text

Description automatically generated

Повече информация за Console.Clear(): <https://docs.microsoft.com/en-us/dotnet/api/system.console.clear?view=net-6.0> .

Сега създайте променлива от тип TimeSpan, която ще ни помогне да създаваме различна **трудност** за нашето приложение. След това трябва да **прочетем входа на потребителя**. Вече знаем как да го направим:



Сега трябва да създадем switch-case конструкция, която да **проверява избора** на потребителя от **менюто**. За всеки случай трябва да **записваме** различните **милисекунди**, в които потребителят трябва да реагира, преди да **изгуби играта**. Трябва да съхраним тези милисекунди в променливата requiredReactionTime. Ще използваме метода TimeSpan.FromMilliseconds():

Table

Description automatically generated with low confidence

Повече информация за TimeSpan.FromMilliseconds(): <https://docs.microsoft.com/en-us/dotnet/api/system.console.clear?view=net-6.0> .

Сега нека да изпълним приложението в конзолата и да проверим дали кодът **работи правилно**:

Text

Description automatically generated

Text

Description automatically generated

Трябва да виждате менюто и да можете да **пишете** в **конзолата**.

#### Задаване на сигнал за потребителя

Сега **изчистете конзолата** отново и създайте **втора променлива** от тип TimeSpan, която ще бъде **сигнал**, който потребителят ще получи. Този сигнал ще му помогне да разбере кога да **натисне „спусъка“ на пистолета**. Като **параметър** трябва да подадем вече създадената променлива **random** и да извикаме нейния Next() метод. Накрая трябва да **отпечатаме** нашата константа Wait на конзолата.

Text

Description automatically generated

След това трябва да създадем **променлива** от тип StopWatch. Тя ще ни помогне да **проследим изминалото време**. Също ще използваме нейния метод Restart(), който ще **рестартира** изминалото време и ще започне да го измерва **отначало**. За да използваме **променлива** от тип StopWatch, трябва да добавим using System.Diagnostics в началото на нашия код (на първия ред):

|  |
| --- |
| **using** System.Diagnostics |

Трябва да изглежда така:

Text

Description automatically generated

Създайте **булева променлива**, която ще използваме малко по-късно. Тя ще покаже дали **потребителят е стрелял твърде бързо**. Първоначално задайте стойността и на **false**:

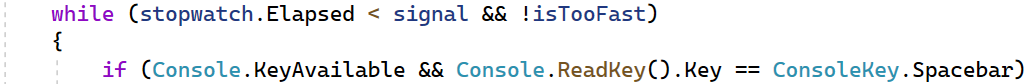


Сега създайте вложен while цикъл. Трябва да **итерира** докато **изминалото време на таймера** стане **повече от сигнала** в TimeSpan променливата **или** нашата **булева променлива** стане равна на **true** (което означава, че потребителят вече е стрелял).

Text

Description automatically generated with low confidence

В **цикъла** трябва да проверим дали някой клавиш на клавиатурата е бил **натиснат** с Console.KeyAvailable и дали този клавиш е конкретно [Spacebar]:



Ако if конструкцията върне **true**, задаваме **Boolean** стойността на true. Вече знаете как се прави това:

Graphical user interface, text, application

Description automatically generated

Повече информация за:

StopWatch: <https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch?view=net-6.0>,

**Console.KeyAvailable**: <https://docs.microsoft.com/en-us/dotnet/api/system.console.keyavailable?view=net-6.0> and **Console.ReadKey()**: <https://docs.microsoft.com/en-us/dotnet/api/system.console.readkey?view=net-6.0>.

Сега нека да изпълним приложението в конзолата и да проверим дали кодът ни **работи правилно**:

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Трябва да виждате началното **меню** в играта. След като въведете **число** за **трудността (1 до 4)**, трябва да видите екрана за изчакване (“wait”).

**Натиснете клавиша**[Spacebar] на клавиатурата си, **докато** сте във **фаза “wait”** – трябва да бъдете **върнати към началното меню** на играта.

#### Имплементиране на логика за победа и загуба

Сега **зачистете конзолата**.



Създайте if-else конструкция, в която ще проверим дали нашата **булева** променлива isTooFast е равна на true. Ако е, отпечатайте на конзолата **константата** loseTooFast. Вече знаете как да направите това:

A picture containing graphical user interface

Description automatically generated

Ако isTooFast е false, трябва да добавим още малко **логика**. Първо, **отпечатайте константата** Fire на конзолата, след това **рестартирайте** stopwatch (хронометъра), като използвате метода method Restart(). Създайте втора **булева** променлива isTooSlow. Последната трябва **първоначално да е** true и ще съхранява информация дали **реакцията на потребителя е твърде бавна**:

Graphical user interface, text, application

Description automatically generated

След това създайте **променлива** от тип TimeSpan, именувайте я reactionTime и задайте нейната стойност на **стойност по подразбиране** чрез ключовата дума default:



Сега създайте **вложен** while **цикъл**, който **итерира** докато **изминалото** време от нашия **хронометър** е повече от стойността на requiredReactionTime или isTooSlow е равна на false:

Text

Description automatically generated with low confidence

В този цикъл проверете дали някой клавиш е бил **натиснат** с Console.KeyAvailable и дали този клавиш е [Spacebar]. Ако if конструкцията върне **true**, задайте isTooSlow на false. Нашата TimeSpan променлива трябва да стане **равна** на **изминалото време**:

Graphical user interface, text, application

Description automatically generated

Сега остава да **проверим** дали потребителят е **спечелил** играта. Първо, **зачистете** конзолата, след това създайте if-else конструкция, която проверява дали променливата isTooSlow е равна на true. Ако е, **отпечатайте** на конзолата нашата константа LoseTooSlow:

Graphical user interface, text

Description automatically generated

В другия случай отпечатайте на конзолата константата Win и **визуализирайте съобщение за времето на реакция на потребителя**:

Graphical user interface

Description automatically generated with low confidence

Нека сега да изпълним приложението в конзолата и да проверим дали нашият код **работи правилно**. Изберете **трудност** и **изчакайте** докато се появи **сигналът** "Fire":

Text

Description automatically generated

Ако натиснете [Spacebar] бутона **достатъчно бързо**, ще **спечелите** играта:

Diagram

Description automatically generated with low confidence

Ако сте **твърде бавни**, ще **загубите**:

Text

Description automatically generated with medium confidence

Сега видяхме, че нашето **приложение** работи правило и можем да видим екраните за "Fire", "Win" и "Lose", но още не сме приключили.

#### Логика за рестартиране на играта

Можете да имплементирате и **логика**, с която потребителят да **избира** дали да **продължи** да играе или да **прекрати** играта:

Graphical user interface, text, application, chat or text message

Description automatically generated

Това е всичко, от което се нуждаем, за да **завършим** нашия **проект**. След като го изпълним, играта трябва да изглежда така:

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

Сега нека я качим в GitHub.

## Качване на проекта в GitHub

Вече знаете как да качите проекта с **Git** **Bash** или **TortoiseGit**.

### Използвайки TortoiseGit (Опция 1)

Използвайте **Git clone** за клониране с TortoiseGit. Отидете в директорията, в която искате да го съхраните, натиснете **десен бутон** в папката и изберете click [Git Clone**]**.

Graphical user interface, application

Description automatically generated

Сега отидете във вашия **GitHub** профил, отворете новосъздаденото **репо** и копирайте неговия **repo URL**.

Отворете TortoiseGit, поставете **URL**-а и кликнете [OK].

Graphical user interface, application

Description automatically generated

Файловете от вашето GitHub репо ще бъдат свалени в **подпапка** с името на проекта в GitHub, в случая "**QuickDrawByPeter**".

Graphical user interface, application

Description automatically generated

Следващото нещо, което трябва да направите, е да **добавите проекта** към **клонираното репо**. Можете да използвате “**Cut & Paste**”. Трябва да изглежда така:

Graphical user interface, text, application

Description automatically generated

Сега нека **качим** промените на проекта в GitHub.

Можем да използваме [GitCommit…]от TortoiseGit. Отидете в папката на проекта, натиснете **десен бутон** и кликнете[Git Commit -> "main"…].

Graphical user interface, text, application

Description automatically generated

Добавете **подходящо** съобщение и кликнете [Add], за да не пропуснете някои файлове, и след това изберете [Commit].

Graphical user interface, application

Description automatically generated

След това кликнете [Push] и после [OK].

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Това е всичко, от което се нуждаете, за да **качите вашия проект** във вашето **GitHub репо**, използвайки TortoiseGit.

### Използвайки Git Bash (Опция 2)

Като **алтернатива на предишния вариант**, ако нямате "**TortoiseGit**", можете да използвате"**Git Bash**", за да качите проекта си във вашето GitHub репо.

Първо, ако нямата Git на своя **компютър**, го **инсталирайте** от <https://git-scm.com/downloads>.

Отидете в **директорията** с проекта, натиснете **десен бутон** в папката и изберете "**Git Bash Here**", за да отворите конзолата на Git. Ако менюто "**Git Bash Here**" **липсва**, първо трябва да инсталирате Git.

Graphical user interface, application

Description automatically generated

Напишете командата **"**gitclone**"**, последвана от линка на **репото**:

|  |
| --- |
| git clone |

Поставете **URL**-а към вашето репо след командата.

Text

Description automatically generated

Файловете от вашето репо ще бъдат свалени в **подпапка** с името на самия проект в GitHub, в нашия случай "**QuickDrawByPeter**".

Graphical user interface, application

Description automatically generated

Следващото нещо, което трябва да направите, е да **добавите файловете от проекта** в **клонираната папка**. Трябва да изглежда така:

Graphical user interface, text, application

Description automatically generated

Сега сме готови да качим нашите промени. Отидете в папката на репото, натиснете **десен бутон**, изберете"GitBashHere" и изпълнете следните **команди**.

|  |
| --- |
| git status |

**Git status** показва състоянието на директорията.

Text

Description automatically generated

Сега напишете:

|  |
| --- |
| git add . |

Горната команда **добавя** всички нови/модифицирани файлове към вашето локално **Git repo**.

След това напишете:

|  |
| --- |
| git commit -m "Uploaded the game Quick draw" |

Тази команда **commit**-ва вашите промени във вашето локално **Git repo**. Също трябва да **добавите** подходящо **съобщение**.

Имаме да изпълним **още две** **команди**. Напишете:

|  |
| --- |
| git pull |

Тази команда **обновява** вашето локално **репо** от GitHub. Сваля последната версия на проекта от GitHub и я слива с вашето локално копие.

Последното нещо, което трябва да направим, е да **push**-нем промените чрез следната команда:

|  |
| --- |
| git push |

Тази команда **качва вашите локални промени в GitHub**.

Text, chat or text message

Description automatically generated

Това е всичко, от което се нуждаете, за да **обновите** вашето **репо**, използвайки **Git Bash**.

## \* Модифициране на кода, добавяне на нови функционалности

Нека да си **поиграем с кода** и **да го модифицираме**.

|  |  |
| --- | --- |
| Icon  Description automatically generated | Това е ваш собствен проект. **Бъдете уникални**.   * Имплементирайте **свои собствени функционалности**. * **Имплементирайте кода сами**, добавете ваш стил, форматиране, коментари и т.н. |

### Добавяне на брояч за победи и загуби

Можете да добавите **брояч за победи и загуби**, така че когато потребителят приключи играта, може да получи определен брой точки.

You can **add a counter for the wins and losses**, so whenever the player finishes the game will receive points.

### Допълнителни идеи

* Можете да помислите за по-интересен начин да **нарисувате анимациите на играта**.

### Качване в GitHub

**Сега качете промените в кода си** във вашето GitHub репо!

|  |  |
| --- | --- |
| Icon  Description automatically generated | Много е важно да **commit-вате често** вашия код в GitHub. По този начин създавате **богата commit история** за вашия проект:  A picture containing chart  Description automatically generated |

## Добавете README.md файл

Силно препоръчително е да напишете **документация за вашия проект в GitHub**, за да опишете за какво е предназначен. Започнете като редактирате файла README.md от вашето репо:

Graphical user interface, text, application, email

Description automatically generated

### Секции на документацията

Добавете **информация** за вашия проект във вашия README.md файл. Обикновено документацията съдържа следните **секции**:

* **Заглавие на проекта** (трябва да отговаря на въпроса "Какво има в проекта?")
* **Цели на проекта**  (какъв проблем решаваме, напр. имплементация на игра)
* **Решение**  (трябва да описва как решаваме проблема 🡪 **алгоритми**, **технологии**, **библиотеки** и т.н.)
* **Линк към source код**
* **Скрийншоти** (скрийншоти към различните функционалности на проекта)
* **Демо наживо** (линк към директно демо на проекта)

### Използвайте markdown

Забележете, че README.md файла се пише на **Markdown език**. Markdown комбинира текст и специално форматирани тагове.

Можете да научите повече за **Markdown** тук: <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>.

### Цели на проекта

Започнете документацията, като опишете **целите на проекта**. Какъв проблем решава проекта?

### Примерна документация

Това е **пример** как може да документирате проекта си. Съветваме ви да не го преписвате, а сами да структурирате документацията си!

Graphical user interface, Teams

Description automatically generated

Можете да добавите **подходящи изображения**, за да подобрите документацията. Можете да добавите изображение по следния начин:



Добавете информация за **входа** и **изхода** на проекта:

Graphical user interface, text, application, email

Description automatically generated

### Вашето решение

Опишете как **решавате** проблема: **алгоритми, технологии, библиотеки и т.н.**

### Линк към сорс код

Добавете **линк** към вашия **сорс код**, както следва:

Logo

Description automatically generated with low confidence

### Скрийншоти

Добавете **скрийншоти** на вашия проект:

1. **Направете скрийншот** с предназначен за това инструмент (например [Snipping Tool](https://support.microsoft.com/en-us/windows/open-snipping-tool-and-take-a-screenshot-a35ac9ff-4a58-24c9-3253-f12bac9f9d44) за Windows).
2. **Поставете** скрийншота във файла за документация, използвайки [Ctrl+V]:

Примерни скрийншоти за играта "Quick Draw":

A picture containing graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

## Качване на приложението си в Replit

Вече би трябвало да имате **Replit** профил. Добавете вашия **проект**, за да можете да показвате **проектите си** на вашите приятели и да сложите линка в секцията "**live demo links**" във вашата GitHub документация.

Създайте **нов проект** в Replit от **менюто** в **горния** **ляв ъгъл**.

Кликнете [Create], след това избере **езика**, на който е **написан** вашият проект, изберете **име** и **създайте** проекта. Изберете "Mono C#".

Graphical user interface, text, application, chat or text message

Description automatically generated

**Поставете вашия код** във файла "Main.cs":

Graphical user interface, text, application

Description automatically generated

**Кликнете** [Run] и ще изпълните приложението си директно в браузъра:

Graphical user interface

Description automatically generated with medium confidence

Сега можете да **споделяте** **приложението си** с ваши приятели.

## Добавете Replit линк към вашия README.md файл

Сега добавете линк към демо от вашия проект към вашата GitHub документацията:

A picture containing text

Description automatically generated

Можете да направите **скрийншот** от Replit.com и **да го поставите** в GitHub документацията директно чрез **[Ctrl+V]**.

След промените, документацията трябва да изглежда така:

Graphical user interface, application

Description automatically generated

Поздравления! Успешно завършихте вашата **трета конзолна игра** и вашия **трети проект** в **GitHub портфолиото ви**.