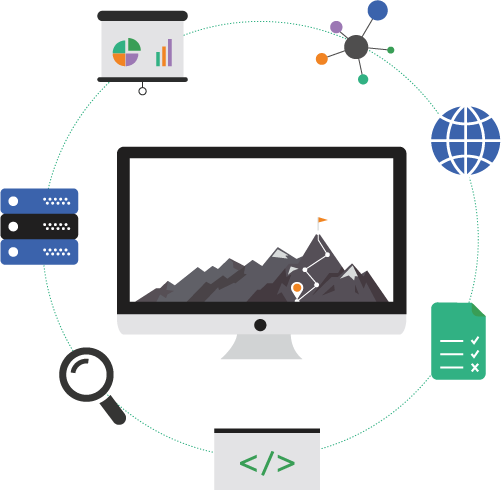
**Skript Münzenjäger**



Dein Weg beginnt

mit der Schnupperlehre im BIT.

Inhalt

[1 Projektcode in Visual Studio öffnen 3](#_Toc136348332)

[2 Spielfeld erstellen. 4](#_Toc136348333)

[3 Informationen zum Spiel 4](#_Toc136348334)

[4 Buttons 6](#_Toc136348335)

[5 Player 7](#_Toc136348336)

[6 Game Timer 8](#_Toc136348337)

[7 Coins 9](#_Toc136348338)

[8 Enemy Bot 10](#_Toc136348339)

[9 Hinderniss 11](#_Toc136348340)

[10 Speed 11](#_Toc136348341)

[11 Enemy 12](#_Toc136348342)

[12 Eigene Ideen 13](#_Toc136348343)

[13 Projekt exportieren 13](#_Toc136348344)

[14 Wie du Münzenjäger zu Hause ausführst 14](#_Toc136348345)

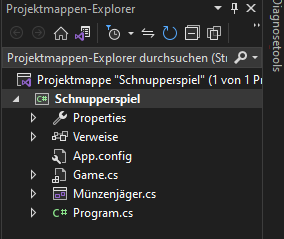
# Projektcode in Visual Studio öffnen

Starte jetzt Visual Studio und klicke auf «*Projekt oder Projektmappe öffnen»*.

Suche danach dein Projekt auf dem Desktop. Wenn du es gefunden hast, musst du die Datei «Schnupperspiel.snl» auswählen. (Die Datei befindet sich im Ordner «Schnupperprojekt»)

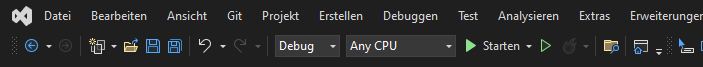
Links siehst du den Projektmappen-Explorer wo du alle Dateien des Projektes sehen kannst. Mach dort ein Doppelklick auf «Schnupperspiel». Falls «.snl» nicht steht, musst du die Datei suchen, welche wie ein Fenster aussieht und ein violettes Zeichen hat.

Warte bis alle Dateien geladen sind. Wenn der Code noch nicht automatisch geöffnet wurde, kannst du ihn rechts im Projekt-Explorer öffnen. Drücke die Taste F7 auf die Datei «Münzenjäger.cs» wenn du nur ein weisses Fenster statt code siehst. In dieser Datei wirst du während des Tages dein eigenes Spiel programmieren.



**Spiel starten**

Wenn du alles wie auf dem Bild eingestellt hast, kannst du das Spiel starten, indem du auf den grünen Pfeil klickst:



Es kann lange dauern, bis das Spiel das erste Mal fertig geladen hat. Das leere Spiel hat fertig geladen, wenn ein neues Fenster in deinem Desktop geöffnet wird. Das Fenster sollte weiss aussehen mit der Oberschrift «Spielfeld» oben links.

# Spielfeld erstellen.

Als erstes erstellen wir das Spielbrett. In der loadGame() Funktion stehen schon ein paar Zeilen von Code die auskommentiert wurden. Wenn sie auskommentiert sind, sind die Zeilen grün und werden nicht ausgeführt. Damit diese Zeilen ausgeführt werden, musst du bei diesen 4 Zeilen die beiden Schrägstriche («//») vorne an der Zeile entfernen.

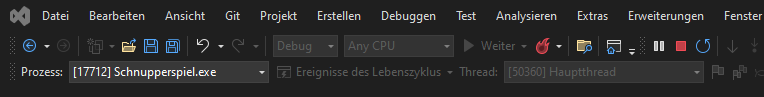
Die zwei letzten Funktionen sind aber noch rot unterstrichen. Das heisst, dass irgendetwas noch nicht stimmt. Beide Funktionen brauchen Parameter um die Grösse und Farbe des Spielbretts zu setzten.  
Setz die Breite auf 800 und die Höhe auf 500. Die Farbe kannst du auf Schwarz setzen (0,0,0).

Zuletzt musst du das Spielbrett zum Spielfeld hinzufügen. Füge dazu folgende Zeile unter der Farbsetzung ein:

game.setPanel(gamePanel);

**Achtung!** Alles was du in der Funktion «loadGame» schreibst muss oberhalb der Zeile game.makeGame(this); stehen. Was du unterhalb dieser Zeile schreibst wird nicht ausgeführt.

Statt jedes Mal das Spielfenster zu schliessen und das Spiel wieder zu starten, kannst du ohne das Fenster zu schliessen auf dem Reload-Knopf drücken (unten markiert). Das Fenster wird dann automatisch geschlossen und mit deinen Änderungen wieder geöffnet.



Jetzt sollte dein Spielfenster nicht mehr leer aussehen. Wenn du ein dunkler Rechteck in deinem Spielfenster siehst kannst du weiter fahren.

# Informationen zum Spiel

Wir müssen aber noch Informationen zum Spiel zeigen können, wie Name, Punkte, Zeit und Highscore. Damit wir diese in unserem Spiel anzeigen können müssen wir zuerst eine Beschriftung für den Informationen erstellen. Mit der folgenden Zeile kannst du ein Label erstellen:

Label labelName = new Label();

labelName ist der Name der Variable und kann irgendwie heissen. Wichtig ist, dass du nie zwei Variablen den gleichen Namen gibst und das der Name aussagekräftig ist, damit du ihn später nicht mit einer anderen Variable verwechselst.

Beim Label setzen wir die Position und Grösse mit den jeweiligen Funktionen (labelName ist der Name der Variable deren Werte du setzen willst):

labelName.setPosition(820,480);

labelName.setSize(250,55);

Als nächstes setzen wir den Text, das angezeigt wird mit:

labelName.setText("Münzenjäger:");

Innerhalb der Hochzeichen ("") kannst du das Reinschreiben, was angezeigt werden soll.

Wir haben jetzt das Label erstellt aber ihn noch nicht zum Spielfeld hinzugefügt. Zuletzt müssen wir also das Label zum Spielfeld hinzufügen. Das Label ist ein Objekt welches wir nie wieder verändern müssen. Daher können wir es mit dieser Code Zeile hinzufügen:

game.add(labelName);

Probiere weitere Labels für die anderen Informationen zu erstellen. Die empfohlene Werte sind in dieser Tabelle:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | X Position | Y Position | Breite | Höhe | Text |
| time | 820 | 10 | 220 | 55 | Time: |
| points | 820 | 70 | 220 | 55 | Points: |
| highscore | 820 | 130 | 220 | 55 | Highscore: |

Die Beschriftungen der Werte haben wir jetzt. Nun müssen wir noch die eigentlichen Werte von time, points und highscore anzeigen.

Die Werte erstellen wir ähnlich wie die Beschriftungen. Statt Labels verwenden wir für die Werte das Objekt Text.

Text textName = new Text();

Auch hier setzen wir die Position und Grösse. Die Funktionen heissen gleich wie bei den Labels.

Weil wir die Werte während dem spiel aktualisieren möchten, müssen wir diese speziell zum Spielfeld hinzufügen. Für den verschiedenen Werten werden folgende Funktionen benutzt um es zum Spielfeld hinzuzufügen:

addTimeText(textName)

addPointsText(textName)

addHighscoreText(textName)

In unserem Spiel haben wir diese Werte für die Position und Grösse verwendet:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | X Position | Y Position | Breite | Höhe |
| time | 1050 | 10 | 94 | 44 |
| points | 1050 | 70 | 94 | 44 |
| highscore | 1050 | 130 | 94 | 44 |

# Buttons

Damit wir das Spiel starten können brauchen wir einen Startknopf. Diesen erstellen so:

Button buttonName = new Button();

Auf dem Button führen wir folgende Funktionen aus um die entsprechenden Werte zu setzen:

setPosition()

setSize()

setColour()

setText()

Wir haben diese Funktionen schon mal für andere Objekte gebraucht.

Unsere Standardwerte für den Startknopf sind:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | Y | Breite | Höhe | Rot | Grün | Blau | Text |
| 12 | 550 | 181 | 62 | 255 | 255 | 255 | Start |

Wir müssen noch festlegen was passiert, wenn der Button gedrückt wird. Das machen wir mit dieser Zeile:

buttonName.Click += new System.EventHandler(game.btnStart\_Click);

Ein Button verändert sich eigentlich während einem Spiel nicht. Probiere selber herauszufinden, wie du den Button dem Spiel hinzufügen musst. Wenn du die früheren Aufträge ansiehst, wirst du die Antwort auch finden.

Probiere für den Stop-Button die richtigen Funktionen zu finden. Lasse die Zeile mit dem EventHandler vorerst beim Knopf weg.

Bei uns hat der Button diese Werte

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | Y | Breite | Höhe | Rot | Grün | Blau | Text |
| 423 | 550 | 181 | 62 | 255 | 255 | 255 | Stop |

Damit man den Stop Knopf nicht drücken kann, während kein Spiel läuft, müssen wir diese Zeile ausführen:

buttonName.Enabled = false;

Mit dieser Zeile geben wir dem Stop Button seine Funktion:

buttonName.Click += new System.EventHandler(game.btnStop\_Click);

**Achtung!** Beim drücken des Start-Buttons stürzt die Applikation ab, da wir noch kein Spieler haben. Dies werden wir in der nächsten Aufgabe lösen.

# Player

Wir schreiben jetzt eine Funktion, die uns ein Player-Objekt zurückgibt. So können wir etwas Platz in der Funktion loadGame sparen. Schreibe also folgende Funktion ausserhalb der Funktion loadGame aber immer noch innerhalb der Klasse.

private Player createPlayer()

{

Player player = new Player();

return player;

}

Führe oberhalb der return player; Zeile die folgenden Funktionen auf player aus:

setSize()

setSpeed()

setPosition()

Gib bei setPosition die variablen xPlayer, yPlayer anstelle von absoluten Werten.

Sonst haben wir diese Werte:

|  |  |  |
| --- | --- | --- |
| Breite | Höhe | Geschwindigkeit |
| 50 | 50 | 4 |

Das Problem besteht weiter hin, weil wir den Spieler noch nicht dem Spiel hinzugefügt haben. Dies machen wir wieder in der loadGame Funktion. Der Spieler gehört aufs Spielbrett (Panel) füge den Spieler mit der add() Funktion und der neu erstellten createPlayer() Funktion dem Panel hinzu.

Jetzt sehen wir den Spieler zwar aber bewegen können wir ihn im Moment nicht.

Füge in der loadGame Funktion diese Zeile hinzu:

KeyDown += new KeyEventHandler(movePlayer);

Anschliessend kannst du in der Datei die movePlayer Funktion suchen.  
Darin ist viel auskommentiert entferne zuerst /\* am Anfang und \*/ am Ende der Funktion.

Schreibe anschliessend in die richtigen if-Statements folgende Funktionen des Players:

moveRight()

moveLeft()

moveUp()

moveDown()

Dies Bewegt den Player noch nicht. Setze darin eine neue Position für den Player. Die neuen werte kannst du mit den Funktionen getPositionX() und getPositionY() vom Player holen.

# Game Timer

Momentan beginnt unser Timer bei 0. Wir wollen aber, dass er bei einer positiven Anzahl Sekunden beginnt. Wenn der Timer bei 0 ankommt soll das Spiel beendet werden.

Dafür müssen wir uns den Timer vom Spiel holen und anschliessend ein Eventhandler hinzufügen. Dies machen wir mit folgenden Code-Zeilen, die wir in loadGame einfügen:

Timer tmrGame = game.tmrGame;

tmrGame.Tick += new System.EventHandler(this.tmrGame\_Tick);

Entferne /\* und \*/ am Anfang und Ende der Funktion tmrGame\_Tick ganz Unten in der Datei.

Anschliessend führen wir in loadGame noch diese Funktionen vom Objekt game aus:

setTime() Welches die Spielzeit festlegt

setTimerGameInterval() welches den Intervall festlegt (wir haben 1000).

In tmrGame\_Tick schreiben wir jetzt das es stoppen soll, wenn die Zeit null ist.

Schreibe dafür ein if-Statement. Den Wert der Zeit bekommst du mit der Funktion von game getTime() und die Vergleichs-Operatoren kannst du im Cheat-Sheet nachschauen.

Wenn du dann im Statement die Funktionen von game timeIsUp() und stopGame() aufrufst wird das Spiel gestoppt.

# Coins

Erstelle eine Funktion, die nichts zurückgibt (Rückgabetyp void), in der wir unsere Coins erstellen.

Erstelle in der Funktion ein Coin. Setze die Grösse auf (20, 20) und füge den neuen Coin der Coin-Liste hinzu mit:

coin.addToList(game.getCoinList());

Die Coin-Liste wird automatisch dem Panel hinzugefügt also musst du das nicht mehr machen.

Für die Position setzen wir in der Klasse am besten oberhalb von loadGame, zwei variablen. Brauche diese Variablen als Ersatz für die x und y Position in der setPosition Funktion. Du musst zusätzlich auch noch das gamePanel mitgeben. Bei einer Variabel direkt in der Klasse muss man auch private, protected oder public mitgeben.

Hole dir vom Game einen nächsten Timer am besten wieder in loadGame. Der Timer heisst tmrCoin.

Erstelle eine Funktion (In den klammern der Funktion muss (object sender, EventArgs e) stehen) dies ist eine EventHandler Funktion.

Schreibe folgende Zeile unterhalb dem neuen Timer in loadGame. Achte darauf, dass du timerName mit dem Namen des Timers auswechselst und EventHandlerName mit der neuen Funktion.

timerName.Tick += new System.EventHandler(this.EventhandlerName);

In der neuen Funktion erstellen wir einen While-Loop. Es soll so lange wiederholt werden bis eine gewisse Anzahl an Coins erstellt wurden. Die Zahl von erstellten Coins bekommst du so:

game.getCoinList().Count

Setze deine zwei Variablen, welche du bei deiner Funktion zum Erstellen von Coins für die Position Setzung brauchst auf random Zahlen. Random zahlen innerhalb des Spielfelds bekommst du so:

random.Next(20, gamePanel.getWidth() - 40)

random.Next(20, gamePanel.getHeight() - 40)

Bevor wir den Coin zur Liste hinzufügen, müssen wir noch überprüfen, dass sich die Coins nicht überlappen würden, also ob die zufällige Position frei ist. Die checkCoinPosition() Funktion von game gibt uns den Wert true zurück wenn an der gegebenen Position noch kein Coin ist.

Im Moment können wir unsere Coins nicht aufsammeln. Dies kannst du mit der LookForCoin Funktion von game machen. Bei dieser Funktion musst du noch mitgeben wieviel ein Coin wert ist. Bei unserem Beispiel wären das 10. Schreibe dies vor oder nach dem While-Statement.

Damit wir unseren Punktestand sehen können müssen wir noch der setScore Funktion von game die Anzahl Punkte mitgeben, welche wir mit der getPoints Funktion von game bekommen.

Damit der Highscore am Ende vom Spiel aktualisiert wird müssen wir zunächst eine neue Funktion machen und diese in tmrGame\_Tick im If-Statement aufrufen.

In der neuen Funktion müssen wir den Highscore setzen, wenn der Punktestand höher ist, als der Highscore. Du wirst diese Funktionen von game brauchen

setHighscore

getHighscore

getPoints

# Enemy Bot

Fügen wir jetzt ein «Enemy» hinzu. Als erstes brauchen wir wieder einen Timer von game welcher tmrEnemy heisst schreib dafür wieder eine EventHandler-Funktion. Schreibe auch eine Funktion die ein EnemyBot erstellt. Hier musst du wie beim Player zusätzlich zu Position und Grösse auch den Speed setzen. Zum Panel wird er automatisch hinzugefügt.

Schreibe eine Funktion, die den Bot bewegt und rufe sie im neuen EventHandler auf.

In der neuen Funktion schreiben wir folgende Zeilen:

foreach (EnemyBot bot in gamePanel.getEnemyBots()){

}

In den klammern schreiben wir ein einfaches If-Else-Statement. In die Kondition schreiben wir den Namen einer neuen Variabel mit dem Datentyp Boolean, die wir in der Klasse neu definieren und auf true setzen. Die Variable wird uns helfen zu bestimmen in welcher Richtung sich unser EnemyBot bewegen soll.  
Auf bot können die Funktionen moveLeft und moveRight ausgeführt werden um den Bot zu verschieben. Schreibe zusätzlich im Foreach-Statement zwei If-Statements, die den Boolean auf true beziehungsweise false setzen, wenn der Bot an einem gewissen Ort ist. Im Beispiel haben wir, wenn der Abstand zwischen dem Rand des Panels und dem Bot über 800 ist und wenn der Abstand zwischen dem Rand des Panels und dem Bot kleiner als seine XPosition ist. Den Abstand zur Wand links kannst du mit der Funktion getLeft von EnemyBot erfahren.

Wenn wir im Moment das Spiel starten, sehen wir noch kein Gegner. Schreibe in tmrGame\_Tick ein If-Statement damit bei einem gewünschten Zeitpunkt ein enemyBot erstellt wird. Die Funktion zum Erstellen haben wir ja schon. Schreibe noch ein weiteres If-Statement damit nicht mehrere enemyBots erstellt werden. Die Anzahl von enemybots die bereits erstellt wurden bekommst du mit getEnemyBots().Count von gamePanel.

Zuletzt musst du noch checken ob der Bot den Player berührt und wenn ja, soll das Spiel stoppen. Mache dies auch in der Funktion, in welcher du den bot bewegst. Die colidesWith Funktion nimmt ein EnemyBot oder Player entgegen und kann auch auf ihnen ausgeführt werden. Den Player bekommst du mit der Funktion getPlayer vom gamePanel und das Spiel kannst du mit der stopGame Funktion von game stoppen.

# Hinderniss

Als nächstes fügen wir ein Hindernis ein. Erstelle in loadGame eine Wall und füge sie dem Panel hinzu. Wir haben in unserem Beispiel diese Werte genommen:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | y | Breite | Höhe | Rot | Grün | Blau |
| 431 | 111 | 30 | 147 | 255 | 255 | 255 |

Im Moment können wir noch durch die Wand hindurch. Dies müssen wir in der movePlayer Funktion ändern. Die Funktion checkWallRight von game (Die ein player, enemy oder enemyBot und ein integer welcher der Speed des mitgegebenen Objekts sein soll entgegennimmt) gibt true zurück wenn das mitgegebene Objekt sich rechts in die Wand bewegen würde. Ergänze die If-Statements damit sich der Player nicht in die Wand bewegen kann. Diese Funktionen funktionieren ähnlich zu der vorher genannten.

checkWallLeft, checkWallTop und checkWallBottom.

# Speed

Geben wir dem Spieler eine neue Funktion. Schreibe dazu in movePlayer ein weiteres If-Statement, welches bei einem Tastendruck eine neue Funktion aufruft, welche dem Spieler einen Speed-Boost gibt.

Führe folgende Funktionen von game in der neuen Funktion aus: setAddedPlayerSpeed, setTimerSpeedInterval, setSpeedTime, setSpeedDelay und wenn du alles gesetzt hast kannst du die speed Funktion ausführen damit der Player für die gesetzte Zeit schneller wird. In unserem Beispiel haben wir diese Werte:

|  |  |  |  |
| --- | --- | --- | --- |
| Added Speed | Intervall | Time | Delay |
| 5 | 1000 | 5 | 10 |

Im Moment wird der Speed nie wieder abgezogen und wir können mehrmals die Taste drücken um unendlich viel Speed zu bekommen. Um das zu ändern, erstelle in der Classe einen Boolean welcher angeben sollte ob Speed hinzugefügt wurde. Danach füge dem If-Statement in movePlayer hinzu, dass wenn Speed hinzugefügt wurde dies nicht ausgeführt wird. Vergiss nicht, den Boolean auf true zu setzen, wenn du Speed hinzugefügt hast.

Damit wir nach einer gewissen Zeit den Speed wieder wegnehmen können müssen wir den tmrSpeed von game holen und einen neuen eventHandler erstellen. (In der EventHandler-Funktion muss in den Klammern (object sender, EventArgs e) stehen) Frage im EventHandler ab, ob die Zeit 0 ist und wenn ja setze den Player Speed auf den Speed minus den hinzugefügten Speed. Die Zeit bekommst du mit der getSpeedTime Funktion, den Spieler mit getPlayer, den Speed mit getSpeed, den hinzugefügten Speed mit getAddedPlayerSpeed (von game) und den Speed kannst du mit setSpeed setzen.

Damit wir den Speed Boost wieder aktivieren können müssen wir nachdem das delay abgelaufen ist den Boolean wieder auf false setzen. Führe dann auch diese Zeile aus:

game.tmrSpeed.Enabled = false;

Das delay ist abgelaufen, wenn die Zeit gleich ist wie der Wert dieser Funktion von game getSpeedDelay.

# Enemy

Als letztes kümmern wir uns um den Enemy der uns verfolgen wird. Erstelle zuerst wieder eine Funktion die einen Enemy erstellt. (setPosition, setSize und setSpeed)

Unsere Werte sind:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Y | Breite | Höhe | Speed |
| 0 | 400 | 50 | 50 | 1 |

Probiere in tmrGame\_Tick den Enemy zu nicht gerade am Anfang zu erstellen, sondern vielleicht 10 Sekunden nach dem Spielstart.

Schreibe eine letzte Funktion, die du auch im eventHandler aufrufst. In der Funktion musst du als erstes den Enemy bekommen. Schreibe ein ähnliches Foreach-Statement, wie das Foreach-Statement der EnemyBots. Ersetze die Funktion getEnemyBots mit getEnemys. Schreibe im Foreach-Statement die If-Statement damit der Enemy dem Spieler folgt. Diese Funktionen könntest du dafür brauchen:

checkWall (Bottom, Top, Right, Left)

move (Up, Down, Right, Left)

getSpeed

getTop (distanz zwischen Objekt und oberen Rand des Panels)

getLeft (distanz zwischen Objekt und dem linken Rand des Panels)

colidesWith

stopGame

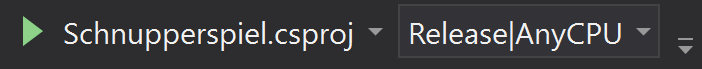
# Eigene Ideen

Falls du noch eine eigene Idee hast, kannst du diese jetzt umsetzen. Du bist nämlich jetzt mit dem eigentlichen Spiel fertig. Du kannst zum Beispiel mehrere Gegner, zufällige Spawn-Koordinaten, u.s.w. implementieren.

# Projekt exportieren

Als letztes kannst du dein Spiel exportieren, damit du es zu Hause auch spielen kannst.

**Anleitung:**

1. Ändere rechts neben der Startdatei folgendes:

**Formatierung**

Von Debug|AnyCPU zu Release|AnyCPU

1. Starte nun das Spiel mit dem grünen Pfeil.
2. Öffne den Explorer und gehe in folgendes Verzeichnis:

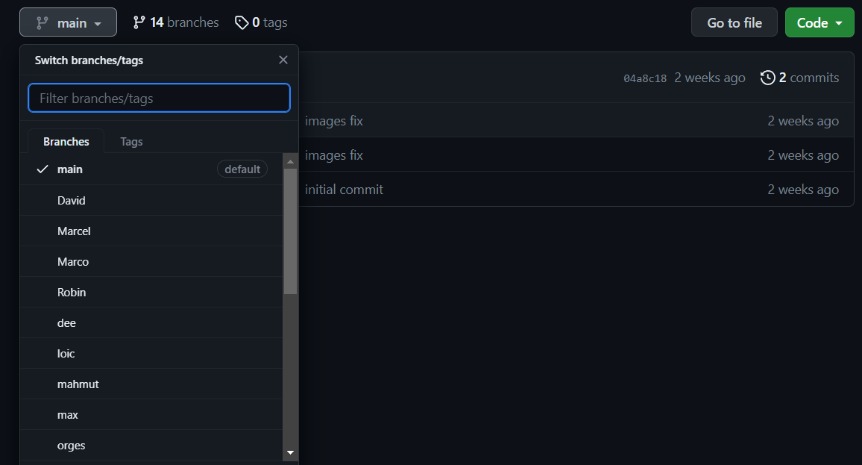
schnupperprojekt\Schnupperspiel\bin\Release

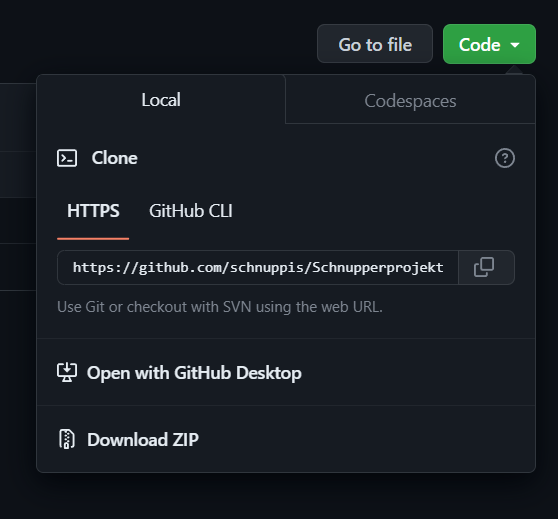
1. In diesem Ordner befindet sich nun dein Spiel: Schnupperspiel.exe 

# Wie du Münzenjäger zu Hause ausführst

Öffne die Webseite des Repositories auf Github

https://github.com/schnuppis/Schnupperprojekt

1. Klicke oben links auf den Knopf, welcher hier auf dem Bild zu sehen ist und wähle beim Dropdown deinen Namen aus. Jetzt sollte bei dem Knopf anstatt «main», dein Name stehen.

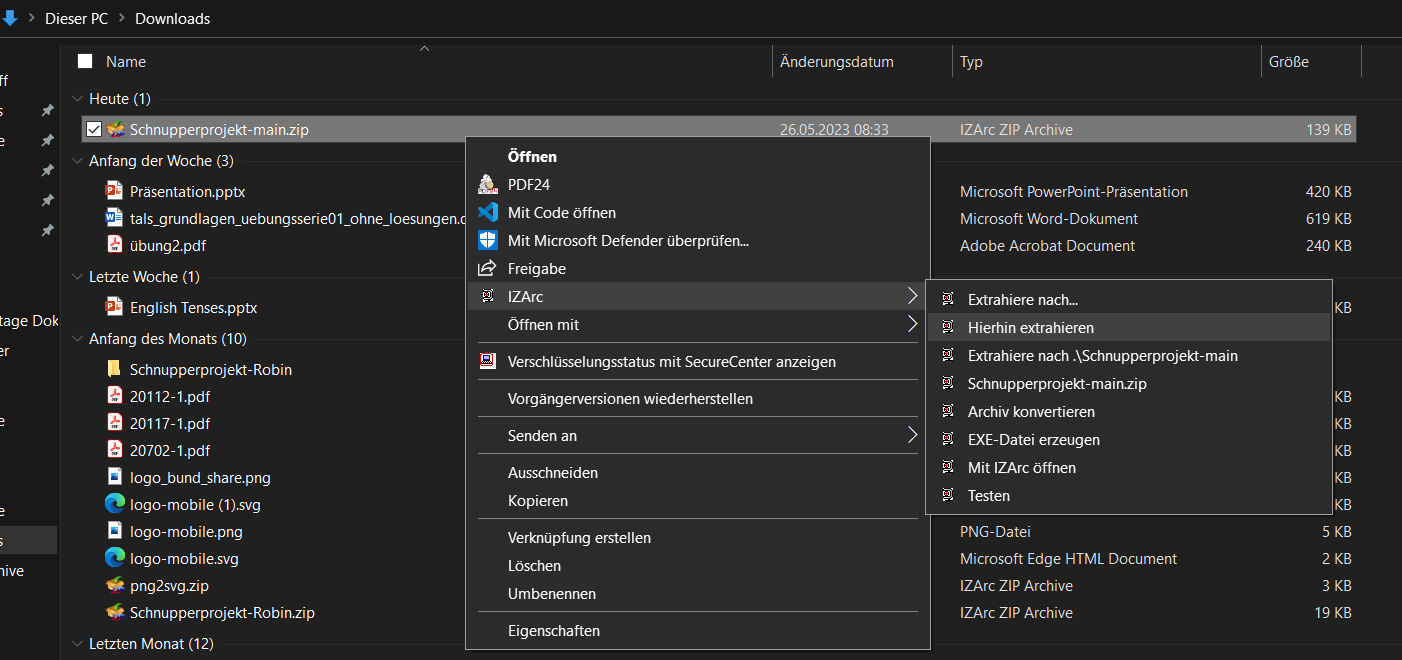


1. Klicke anschliessend oben rechts auf den grünen Knop «Code» und wähle dort die Option «Download ZIP» aus.
2. Visual Studio herunterlanden

Auf der offiziellen Microsoft-Webseite unter «Visual Studio 2022» «Community» auswählen und auf vollständigen Download warten

<https://visualstudio.microsoft.com/downloads/>

1. Sobald Visual Studio installiert ist, kannst du im Windows Explorer zu dem heruntergeladenen Projekt (ZIP-Ordner) navigieren (normalerweise im Downloads-ordner) und das Projekt Entzippen/ Extrahieren.





1. Zum Schluss musst du im extrahierten Projekt die Datei «Schnupperspiel.sln» doppelklicken und mit Visual Studio öffnen. Jetzt öffnet sich das Projekt in Visual Studio und du kannst die Anwendung wie gewohnt starten oder bearbeiten.

Hurra! Du bist am Ziel angekommen! Wir hoffen es hat dir gut gefallen bei uns!



Im Namen der Berufsbildung BIT  
wünschen wir dir alles Gute auf dem  
weiteren Weg der Berufserkundung!

