Smalltalk Programming
Aimbot!

Below is code that you can use to add a much desired ability in the shooter game world – an aimbot! Most everyone who wants this only has the ability to pay money to have it in their game. Almost no one codes their very own aimbot. Are you up for the coding challenge? If you do it, you will be one of the very few number of people who could ever say that they have coded their very own gaming aimbot.

1. Type and save the code below.

```
EllipseMorph subclass: #Shot
    instanceVariableNames: 'aimbotEnabled aimbotSteps'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'ShooterGame'

Shot>>initialize

    super initialize.
    self color: Color red.
    self extent: 16 @ 16.
    aimbotEnabled := false.
    aimbotSteps := 800

Shot>>aimbotEnabled: aBoolean

    aimbotEnabled := aBoolean
```

Shot>>**closestTarget**

```
    | targetDistances |
    targetDistances := Dictionary new.
    owner enemies
        do: [:enemy | targetDistances
            at: (self position dist: enemy position)
            put: enemy].
    ^ targetDistances at: targetDistances keys min
```

Shot>>**aimbotTowards:** aTarget

```
    | myPosition targetPosition myStepTime |
    myPosition := self center.
    targetPosition := aTarget center.
    myStepTime := self stepTime.
    aimbotSteps := aimbotSteps - myStepTime.
    self center: ((targetPosition * myStepTime) + (myPosition * (aimbotSteps
- myStepTime))) // aimbotSteps
```

Shot>>**aimbotTarget**

```
    owner enemies ifNotEmpty: [^self closestTarget]
```

```
Shot>>checkContact
    self top < owner top
        ifTrue: [^ self delete].
    self bottom > owner bottom
        ifTrue: [^ self delete].
    self left < owner left
        ifTrue: [^ self delete].
    self right > owner right
        ifTrue: [^ self delete].
    owner enemies
        do: [:enemy | (self bounds intersects: enemy bounds)
                ifTrue: [self hitEnemy: enemy.
                    ^ self]]

Shot>>move
    (aimbotEnabled and: owner enemies notEmpty)
        ifTrue: [self aimbotTowards: self aimbotTarget]
        ifFalse: [self position: self position - (0@5)].
    self checkContact

Morph subclass: #ShooterGame
    instanceVariableNames: 'ship score aimbotActivated'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'ShooterGame'

ShooterGame>>aimbotActivated

    ^ aimbotActivated
```

ShooterGame>>**initialize**

```
super initialize.
self position: 100 @ 100.
self extent: 640 @ 480.
self color: Color black.
self setNameTo: 'Shooter Game'.
aimbotActivated := false.
self initializeStars.
self initializeShip.
self initializeScore.
self initializeEnemies
```

Ship>>**shoot**

```
| shot |
shot := Shot new aimbotEnabled: owner aimbotActivated.
shot position: self topCenter - shot bottomCenter.
owner addMorph: shot
```

ShooterGame>>**toggleAimbot**

```
aimbotActivated := aimbotActivated not.
(self submorphsSatisfying: [:morph | morph isMemberOf: Shot])
   do: [:shot | shot aimbotEnabled: self aimbotActivated]
```

2. Note that the method `submorphsSatisfying:` works the way same here as the two methods `submorphs select:` do together, which you used elsewhere.

ShooterGame>>**handleKeystroke:** anEvent

```
| keyString |
keyString := anEvent keyString asLowercase.
keyString = 'r' ifTrue: [self initializeEnemies].
keyString = 'a' ifTrue: [self toggleAimbot].
ship keystroke: keyString
```

3. If you are “live coding” (making code changes while your game is still running), you will also need to run this expression in an explore morph window. Run the expression “aimbotActivated := false.” on `ShooterGame` in your explore morph window. Remember to select `ShooterGame` before running the expression. If you quit and start a new game this step will not be necessary.

4. Now you are unstoppable! Is there anything else that might be cool to have for your game?

5. Save and Quit your Smalltalk image.