Pyhton3 Games

Build Games with Python

Hello Camps!
Here is a 15 Python Games Source Code pdf for you

By
Curious Programmer

Index

1. Python free Games

1) Features
2) Quick start

2. Games

1) Snake
2) Paint
3) Pacman
4) Cannon
5) Connect
6) Flappy
7) Memory
8) Pong
9) Simon Says
10) Tic Tac Toe
11) Tiles
12) Tron
13) Life
14) Maze
15) Fidget

1. Free Python Games

Free Python Games is an Apache2 licensed collection of free Python games intended for education and fun. The games are written in simple Python code and designed for experimentation and changes. Simplified versions of several classic arcade games are included.

Python is one of the top-five most popular programming languages in the world and available for free from Python.org. Python includes an extensive Standard Library distributed with your installation. The Standard Library has a module called Turtle which is a popular way to introduce programming to kids. All of the games in Free Python
Games are implemented using Python and its Turtle module.

The games run anywhere Python can be installed which includes desktop computers running Windows, Mac OS, or Linux and older or low-power hardware such as the Raspberry Pi.

Each game is entirely independent from the others and includes comments.

1) Features

- Fun to play!
- Simple Python code
- Easy to install
- Designed for education
- Depends only on the Python Standard Library
- Developed on Python 3.7
- Tested on CPython 2.7, 3.4, 3.5, 3.6, and 3.7
- Tested on Windows, Mac OS X, Raspbian (Raspberry Pi), and Linux
- Tested using Travis CI and AppVeyor CI

2) Quickstart

Installing Free Python Games is simple with pip:

```
$ python3 -m pip install freegames
```

Free Python Games supports a command-line interface (CLI). Help for the CLI is available using:

```
$ python3 -m freegames --help
```

The CLI supports three commands: list, copy, and show. For a list of all games run:

```
$ python3 -m freegames list
```

Any of the listed games may be played by executing the Python module from the command-line. To reference the Python module, combine “freegames” with the name of the game. For example, to play the “snake” game run:

```
$ python3 -m freegames.snake
```

Games can be modified by copying their source code. The copy command will create a Python file in your local directory which you can edit. For example, to copy and play the “snake” game run:

```
$ python3 -m freegames copy snake
$ python3 snake.py
```

Python includes a built-in text editor named IDLE which can also execute Python code. To launch the editor and make changes to the “snake” game run:

```
$ python3 -m idlelib.idle snake.py
```

You can also access documentation in the interpreter with Python’s built-in help function:

```
>>> import freegames
>>> help(freegames)
```

2. Games

If you have have the look at the index we are going to code 15 python based games so lets start with the game number one

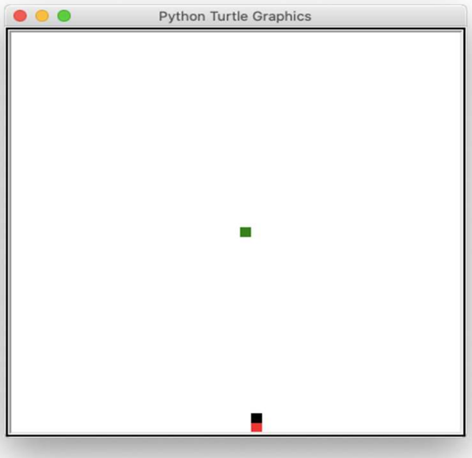
1) Snake

classic arcade game. Use the arrow keys to navigate and eat the green food. Each time the food is consumed, the snake grows one segment longer. Avoid eating yourself or going out of bounds!

Code

```
from random import randrange
from turtle import *

from freegames import square, vector

food = vector(0, 0)
snake = [vector(10, 0)]
aim = vector(0, -10)


def change(x, y):
    "Change snake direction."
    aim.x = x
    aim.y = y


def inside(head):
    "Return True if head inside boundaries."
    return -200 < head.x < 190 and -200 < head.y < 190


def move():
    "Move snake forward one segment."
    head = snake[-1].copy()
    head.move(aim)

    if not inside(head) or head in snake:
        square(head.x, head.y, 9, 'red')
        update()
        return

    snake.append(head)

    if head == food:
        print('Snake:', len(snake))
        food.x = randrange(-15, 15) * 10
        food.y = randrange(-15, 15) * 10
    else:
        snake.pop(0)

    clear()

    for body in snake:
        square(body.x, body.y, 9, 'black')

    square(food.x, food.y, 9, 'green')
    update()
    ontimer(move, 100)


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
listen()
onkey(lambda: change(10, 0), 'Right')
onkey(lambda: change(-10, 0), 'Left')
onkey(lambda: change(0, 10), 'Up')
onkey(lambda: change(0, -10), 'Down')
move()
done()
```

Output :



2) Paint:

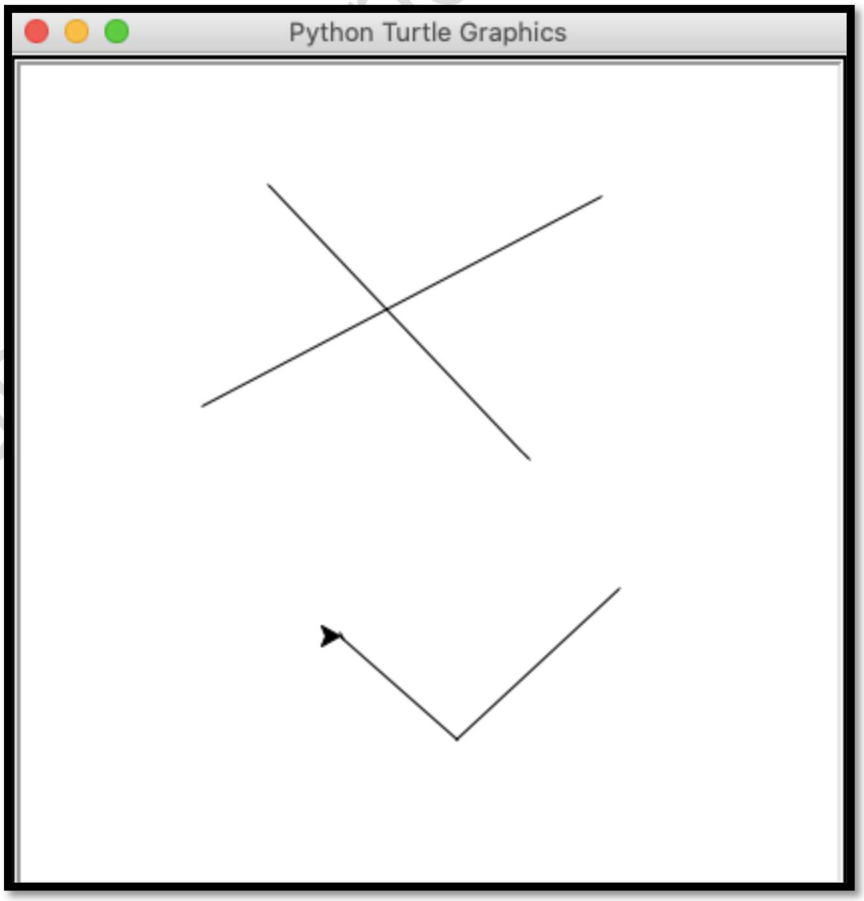
draw lines and shapes on the screen. Click to mark the start of a shape and click again to mark its end. Different shapes and colors can be selected using the keyboard.

Source Code :

```python
from turtle import *

from freegames import vector


def line(start, end):
    "Draw line from start to end."
    up()
    goto(start.x, start.y)
    down()
    goto(end.x, end.y)


def square(start, end):
    "Draw square from start to end."
    up()
    goto(start.x, start.y)
    down()
    begin_fill()

    for count in range(4):
        forward(end.x - start.x)
        left(90)

    end_fill()


def circle(start, end):
    "Draw circle from start to end."
    pass  # TODO


def rectangle(start, end):
    "Draw rectangle from start to end."
    pass  # TODO


def triangle(start, end):
    "Draw triangle from start to end."
    pass  # TODO


def tap(x, y):
    "Store starting point or draw shape."
    start = state['start']

    if start is None:
        state['start'] = vector(x, y)
    else:
        shape = state['shape']
        end = vector(x, y)
        shape(start, end)
        state['start'] = None


def store(key, value):
    "Store value in state at key."
    state[key] = value


state = {'start': None, 'shape': line}
setup(420, 420, 370, 0)
```

```
onscreenclick(tap)
listen()
onkey(undo, 'u')
onkey(lambda: color('black'), 'K')
onkey(lambda: color('white'), 'W')
onkey(lambda: color('green'), 'G')
onkey(lambda: color('blue'), 'B')
onkey(lambda: color('red'), 'R')
onkey(lambda: store('shape', line), 'l')
onkey(lambda: store('shape', square), 's')
onkey(lambda: store('shape', circle), 'c')
onkey(lambda: store('shape', rectangle), 'r')
onkey(lambda: store('shape', triangle), 't')
done()
```

Output



3) Packman

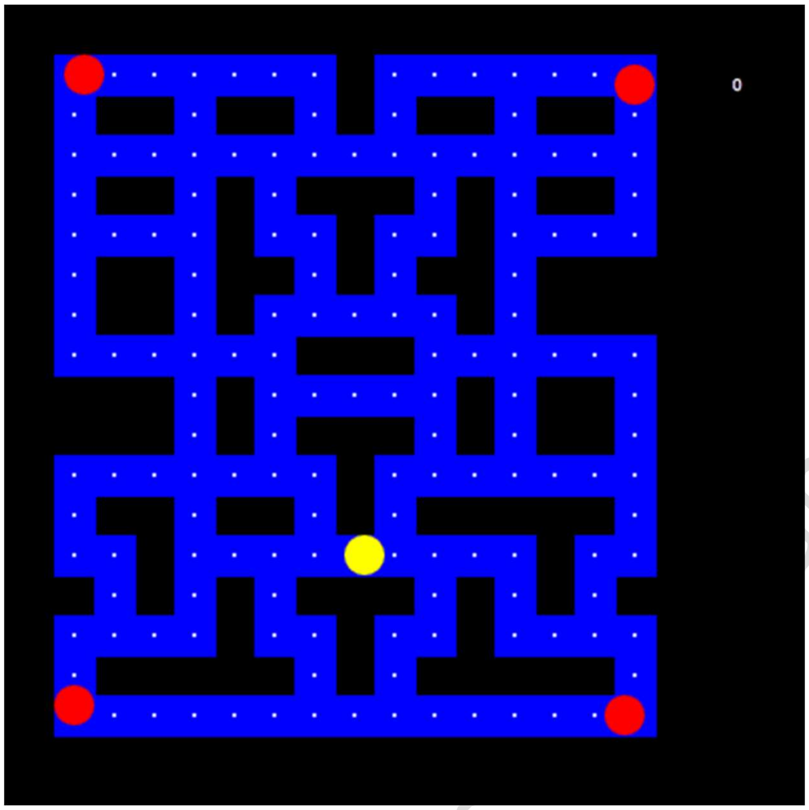
classic arcade game. Use the arrow keys to navigate and eat all the white food. Watch out for red ghosts that roam the maze.

Source code

```
from random import choice
from turtle import *

from freegames import floor, vector

state = {'score': 0}
path = Turtle(visible=False)
writer = Turtle(visible=False)
aim = vector(5, 0)
pacman = vector(-40, -80)
ghosts = [
    [vector(-180, 160), vector(5, 0)],
    [vector(-180, -160), vector(0, 5)],
    [vector(100, 160), vector(0, -5)],
    [vector(100, -160), vector(-5, 0)],
]
```

```python
# fmt: off
tiles = [
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
    0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
]
# fmt: on


def square(x, y):
    "Draw square using path at (x, y)."
    path.up()
    path.goto(x, y)
    path.down()
    path.begin_fill()

    for count in range(4):
        path.forward(20)
        path.left(90)

    path.end_fill()


def offset(point):
    "Return offset of point in tiles."
    x = (floor(point.x, 20) + 200) / 20
    y = (180 - floor(point.y, 20)) / 20
    index = int(x + y * 20)
    return index


def valid(point):
    "Return True if point is valid in tiles."
    index = offset(point)

    if tiles[index] == 0:
        return False

    index = offset(point + 19)

    if tiles[index] == 0:
        return False

    return point.x % 20 == 0 or point.y % 20 == 0


def world():
    "Draw world using path."
    bgcolor('black')
    path.color('blue')

    for index in range(len(tiles)):
        tile = tiles[index]

        if tile > 0:
            x = (index % 20) * 20 - 200
            y = 180 - (index // 20) * 20
            square(x, y)

            if tile == 1:
                path.up()
                path.goto(x + 10, y + 10)
                path.dot(2, 'white')


def move():
    "Move pacman and all ghosts."
    writer.undo()
    writer.write(state['score'])

    clear()

    if valid(pacman + aim):
        pacman.move(aim)

    index = offset(pacman)

    if tiles[index] == 1:
        tiles[index] = 2
        state['score'] += 1
        x = (index % 20) * 20 - 200
        y = 180 - (index // 20) * 20
        square(x, y)

    up()
    goto(pacman.x + 10, pacman.y + 10)
    dot(20, 'yellow')

    for point, course in ghosts:
        if valid(point + course):
            point.move(course)
        else:
            options = [
                vector(5, 0),
                vector(-5, 0),
                vector(0, 5),
                vector(0, -5),
            ]
            plan = choice(options)
            course.x = plan.x
            course.y = plan.y

        up()
        goto(point.x + 10, point.y + 10)
        dot(20, 'red')

    update()

    for point, course in ghosts:
        if abs(pacman - point) < 20:
            return

    ontimer(move, 100)


def change(x, y):
    "Change pacman aim if valid."
    if valid(pacman + vector(x, y)):
        aim.x = x
        aim.y = y


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
writer.goto(160, 160)
writer.color('white')
writer.write(state['score'])
listen()
onkey(lambda: change(5, 0), 'Right')
onkey(lambda: change(-5, 0), 'Left')
onkey(lambda: change(0, 5), 'Up')
onkey(lambda: change(0, -5), 'Down')
world()
move()
done()
```

Output :



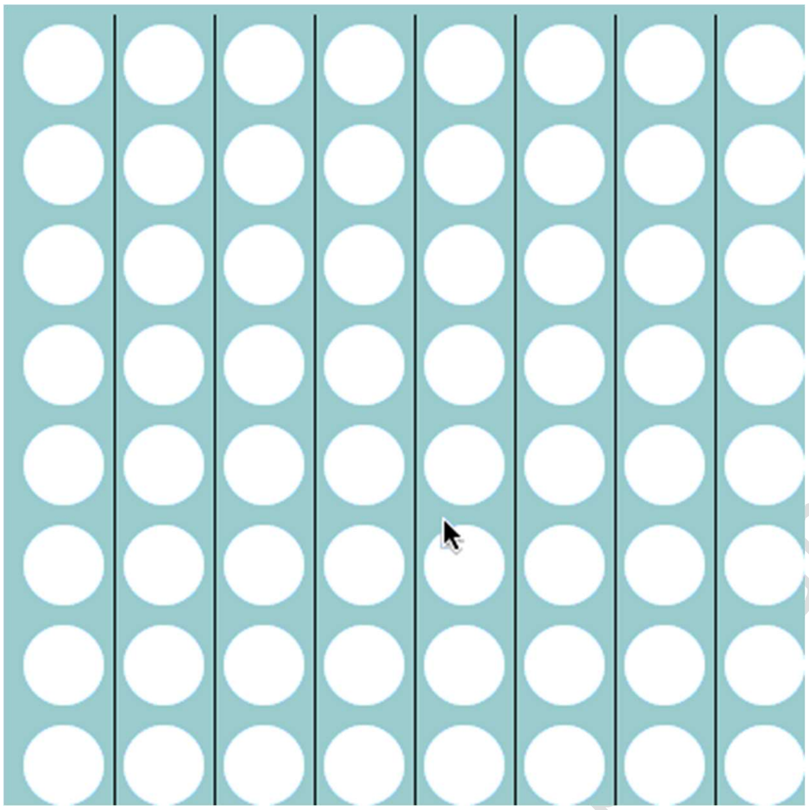
4) Connect

Connect 4 game. Click a row to drop a disc. The first player to connect four discs vertically, horizontally, or diagonally wins!

Source code:

```
from turtle import *

from freegames import line

turns = {'red': 'yellow', 'yellow': 'red'}
state = {'player': 'yellow', 'rows': [0] * 8}


def grid():
    "Draw Connect Four grid."
    bgcolor('light blue')

    for x in range(-150, 200, 50):
        line(x, -200, x, 200)
    for x in range(-175, 200, 50):
        for y in range(-175, 200, 50):
            up()
            goto(x, y)
            dot(40, 'white')

    update()


def tap(x, y):
    "Draw red or yellow circle in tapped row."
    player = state['player']
    rows = state['rows']

    row = int((x + 200) // 50)
    count = rows[row]

    x = ((x + 200) // 50) * 50 - 200 + 25
    y = count * 50 - 200 + 25

    up()
    goto(x, y)
    dot(40, player)
    update()

    rows[row] = count + 1
    state['player'] = turns[player]


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
grid()
onscreenclick(tap)
done()
```

Output :



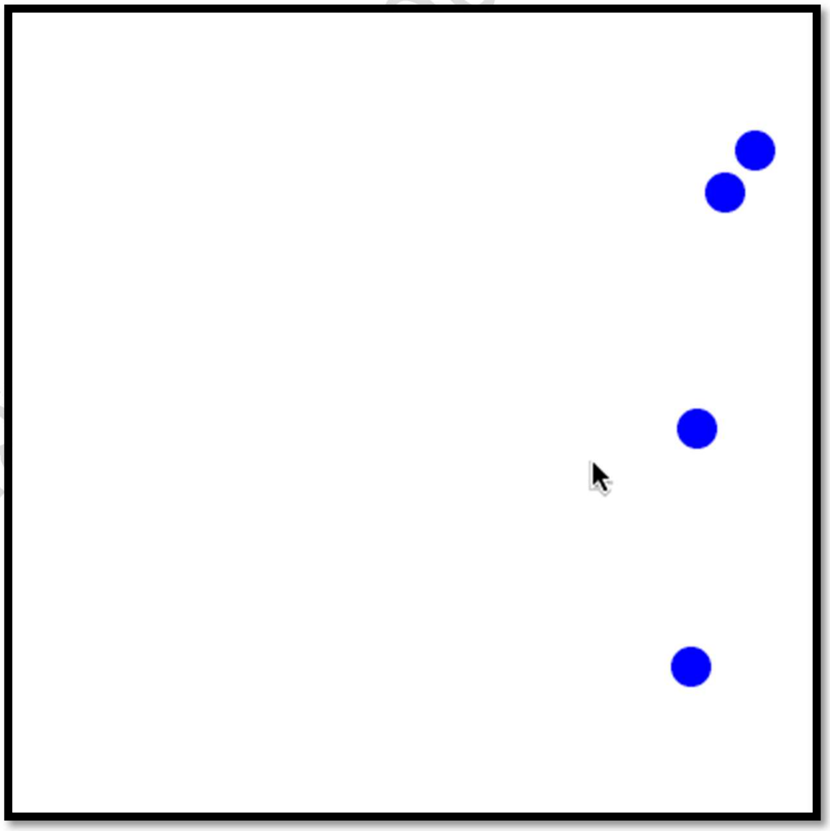
5) Cannon

projectile motion. Click the screen to fire your cannnonball. The cannonball pops blue balloons in its path. Pop all the balloons before they can cross the screen

Source Code:

```
from random import randrange
from turtle import *

from freegames import vector

ball = vector(-200, -200)
speed = vector(0, 0)
targets = []


def tap(x, y):
    "Respond to screen tap."
    if not inside(ball):
        ball.x = -199
        ball.y = -199
        speed.x = (x + 200) / 25
        speed.y = (y + 200) / 25


def inside(xy):
    "Return True if xy within screen."
    return -200 < xy.x < 200 and -200 < xy.y < 200


def draw():
    "Draw ball and targets."
    clear()

    for target in targets:
        goto(target.x, target.y)
        dot(20, 'blue')

    if inside(ball):
        goto(ball.x, ball.y)
        dot(6, 'red')

    update()


def move():
    "Move ball and targets."
    if randrange(40) == 0:
        y = randrange(-150, 150)
        target = vector(200, y)
        targets.append(target)

    for target in targets:
        target.x -= 0.5

    if inside(ball):
        speed.y -= 0.35
        ball.move(speed)

    dupe = targets.copy()
    targets.clear()

    for target in dupe:
        if abs(target - ball) > 13:
            targets.append(target)
    draw()

    for target in targets:
        if not inside(target):
            return

    ontimer(move, 50)


setup(420, 420, 370, 0)
hideturtle()
up()
tracer(False)
onscreenclick(tap)
move()
done()
```

Output:



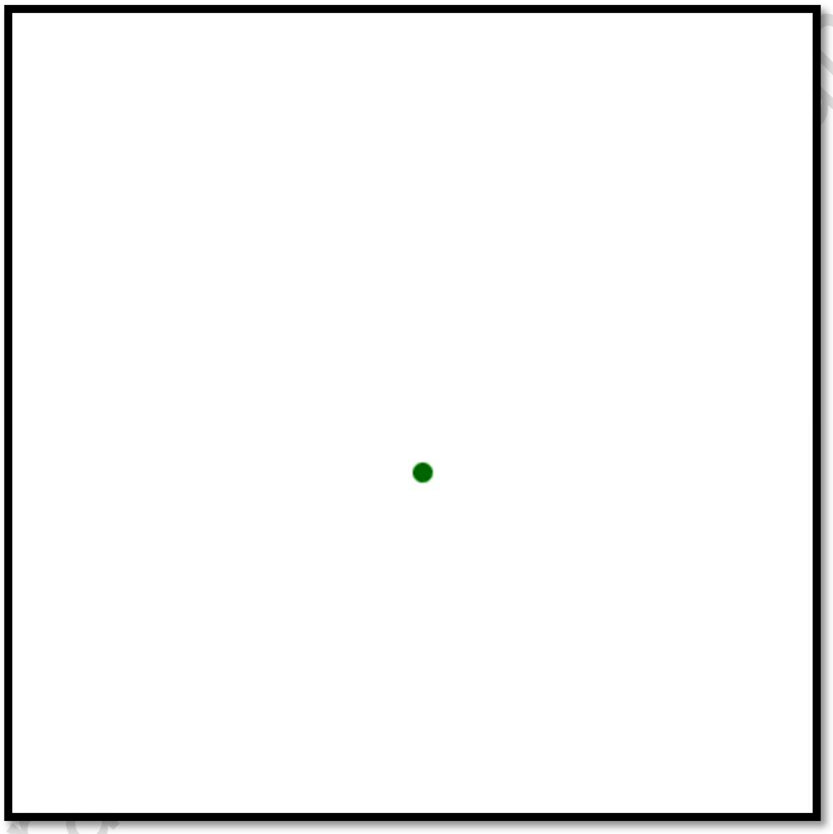
6) Flappy

Flappy-bird inspired game. Click the screen to flap your wings. Watch out for black ravens as you fly across the screen.

Source Code :

```python
from random import *
from turtle import *

from freegames import vector

bird = vector(0, 0)
balls = []


def tap(x, y):
    "Move bird up in response to screen tap."
    up = vector(0, 30)
    bird.move(up)


def inside(point):
    "Return True if point on screen."
    return -200 < point.x < 200 and -200 < point.y < 200


def draw(alive):
    "Draw screen objects."
    clear()

    goto(bird.x, bird.y)

    if alive:
        dot(10, 'green')
    else:
        dot(10, 'red')

    for ball in balls:
        goto(ball.x, ball.y)
        dot(20, 'black')

    update()


def move():
    "Update object positions."
    bird.y -= 5

    for ball in balls:
        ball.x -= 3

    if randrange(10) == 0:
        y = randrange(-199, 199)
        ball = vector(199, y)
        balls.append(ball)

    while len(balls) > 0 and not inside(balls[0]):
        balls.pop(0)

    if not inside(bird):
        draw(False)
        return

    for ball in balls:
        if abs(ball - bird) < 15:
            draw(False)
            return

    draw(True)
    ontimer(move, 50)


setup(420, 420, 370, 0)
hideturtle()
up()
tracer(False)
onscreenclick(tap)
move()
done()
```

Output :



7) Memory

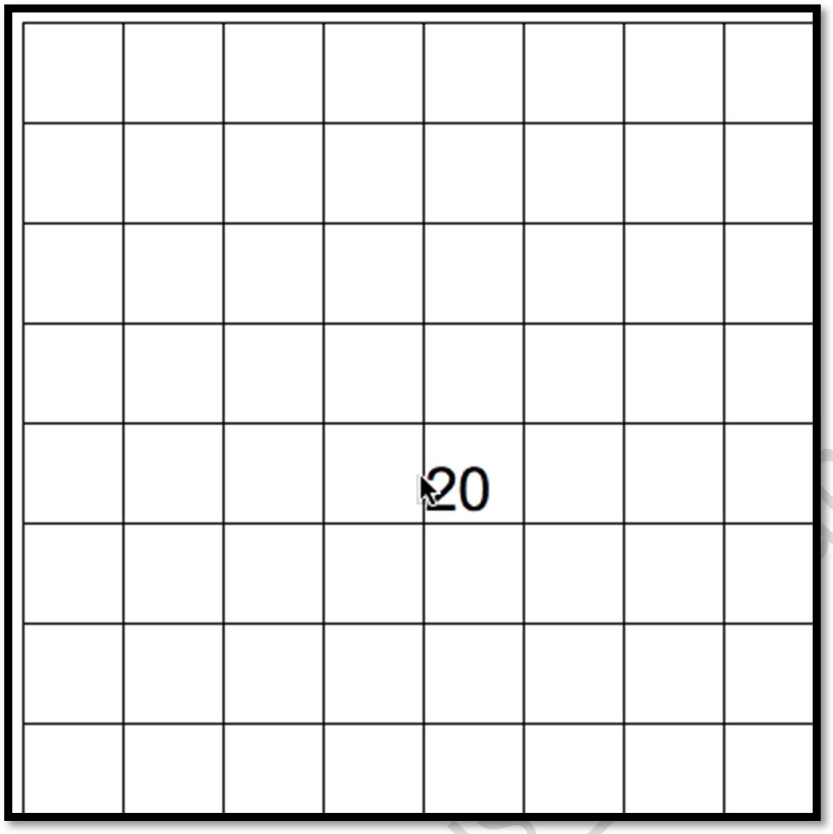
puzzle game of number pairs. Click a tile to reveal a number. Match two numbers and the tiles will disappear to reveal an image.

Source code :

```
from random import *
from turtle import *

from freegames import path
```

```python
car = path('car.gif')
tiles = list(range(32)) * 2
state = {'mark': None}
hide = [True] * 64


def square(x, y):
    "Draw white square with black outline at (x, y)."
    up()
    goto(x, y)
    down()
    color('black', 'white')
    begin_fill()
    for count in range(4):
        forward(50)
        left(90)
    end_fill()


def index(x, y):
    "Convert (x, y) coordinates to tiles index."
    return int((x + 200) // 50 + ((y + 200) // 50) * 8)


def xy(count):
    "Convert tiles count to (x, y) coordinates."
    return (count % 8) * 50 - 200, (count // 8) * 50 - 200


def tap(x, y):
    "Update mark and hidden tiles based on tap."
    spot = index(x, y)
    mark = state['mark']

    if mark is None or mark == spot or tiles[mark] != tiles[spot]:
        state['mark'] = spot
    else:
        hide[spot] = False
        hide[mark] = False
        state['mark'] = None


def draw():
    "Draw image and tiles."
    clear()
    goto(0, 0)
    shape(car)
    stamp()

    for count in range(64):
        if hide[count]:
            x, y = xy(count)
            square(x, y)

    mark = state['mark']

    if mark is not None and hide[mark]:
        x, y = xy(mark)
        up()
        goto(x + 2, y)
        color('black')
        write(tiles[mark], font=('Arial', 30, 'normal'))

    update()
    ontimer(draw, 100)


shuffle(tiles)
setup(420, 420, 370, 0)
addshape(car)
hideturtle()
tracer(False)
onscreenclick(tap)
draw()
done()
```

Output :



8) Pong

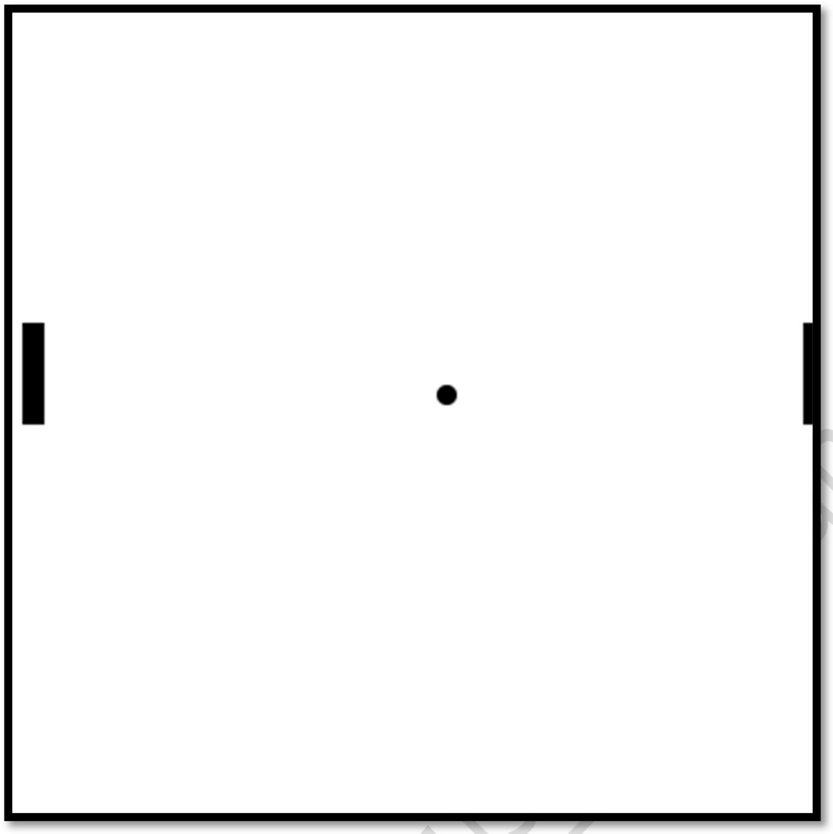
classic arcade game. Use the keyboard to move your paddle up and down. The first player to miss the ball loses.

Source Code:

```
from random import choice, random
from turtle import *

from freegames import vector
```

```python
def value():
    "Randomly generate value between (-5, -3) or (3, 5)."
    return (3 + random() * 2) * choice([1, -1])


ball = vector(0, 0)
aim = vector(value(), value())
state = {1: 0, 2: 0}


def move(player, change):
    "Move player position by change."
    state[player] += change


def rectangle(x, y, width, height):
    "Draw rectangle at (x, y) with given width and height."
    up()
    goto(x, y)
    down()
    begin_fill()
    for count in range(2):
        forward(width)
        left(90)
        forward(height)
        left(90)
    end_fill()


def draw():
    "Draw game and move pong ball."
    clear()
    rectangle(-200, state[1], 10, 50)
    rectangle(190, state[2], 10, 50)

    ball.move(aim)
    x = ball.x
    y = ball.y

    up()
    goto(x, y)
    dot(10)
    update()

    if y < -200 or y > 200:
        aim.y = -aim.y

    if x < -185:
        low = state[1]
        high = state[1] + 50

        if low <= y <= high:
            aim.x = -aim.x
        else:
            return

    if x > 185:
        low = state[2]
        high = state[2] + 50

        if low <= y <= high:
            aim.x = -aim.x
        else:
            return

    ontimer(draw, 50)


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
listen()
onkey(lambda: move(1, 20), 'w')
onkey(lambda: move(1, -20), 's')
onkey(lambda: move(2, 20), 'i')
onkey(lambda: move(2, -20), 'k')
draw()
done()
```

Output:



9) Simon Says

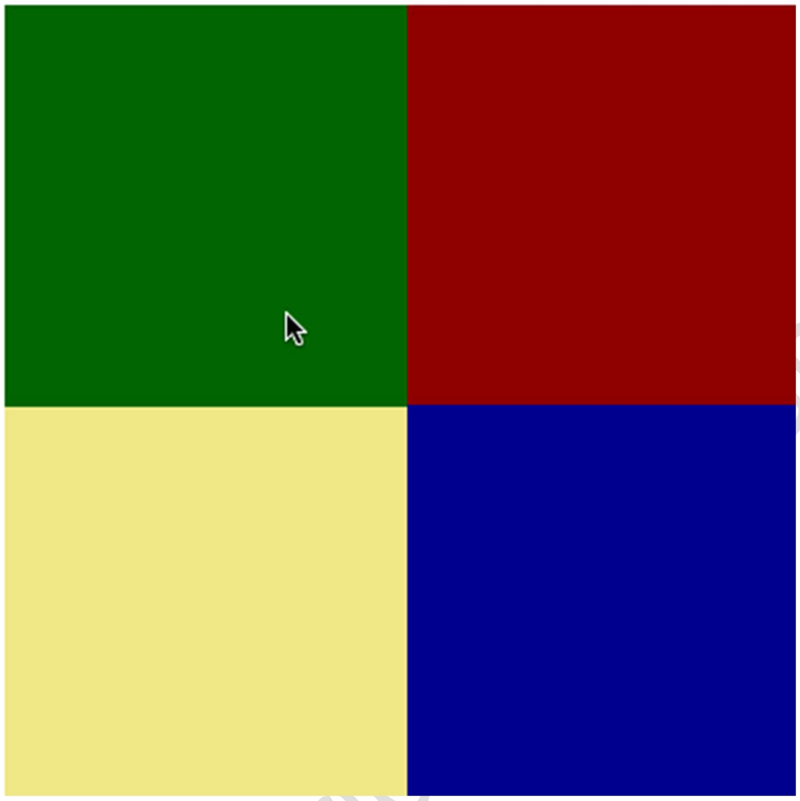
classic memory puzzle game. Click the screen to start. Watch the pattern and then click the tiles in the same order. Each time you get the sequence right the pattern gets one step longer.

Source Code

```
from random import choice
from time import sleep
from turtle import *

from freegames import floor, square, vector

pattern = []
guesses = []
tiles = {
    vector(0, 0): ('red', 'dark red'),
    vector(0, -200): ('blue', 'dark blue'),
    vector(-200, 0): ('green', 'dark green'),
    vector(-200, -200): ('yellow', 'khaki'),
}
```

```python
def grid():
    "Draw grid of tiles."
    square(0, 0, 200, 'dark red')
    square(0, -200, 200, 'dark blue')
    square(-200, 0, 200, 'dark green')
    square(-200, -200, 200, 'khaki')
    update()


def flash(tile):
    "Flash tile in grid."
    glow, dark = tiles[tile]
    square(tile.x, tile.y, 200, glow)
    update()
    sleep(0.5)
    square(tile.x, tile.y, 200, dark)
    update()
    sleep(0.5)


def grow():
    "Grow pattern and flash tiles."
    tile = choice(list(tiles))
    pattern.append(tile)

    for tile in pattern:
        flash(tile)

    print('Pattern length:', len(pattern))
    guesses.clear()


def tap(x, y):
    "Respond to screen tap."
    onscreenclick(None)
    x = floor(x, 200)
    y = floor(y, 200)
    tile = vector(x, y)
    index = len(guesses)

    if tile != pattern[index]:
        exit()
    guesses.append(tile)
    flash(tile)

    if len(guesses) == len(pattern):
        grow()

    onscreenclick(tap)


def start(x, y):
    "Start game."
    grow()
    onscreenclick(tap)


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
grid()
onscreenclick(start)
done()
```

Output :



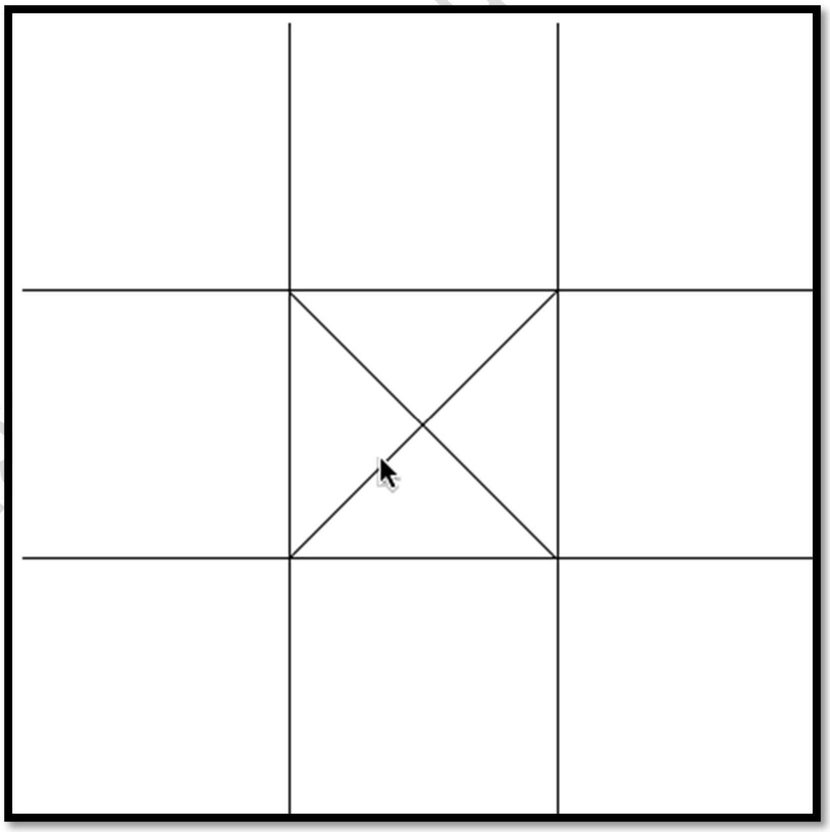
10) Tic Tac Toe

classic game. Click the screen to place an X or O. Connect three in a row and you win!

Source Code

```
from turtle import *

from freegames import line


def grid():
    "Draw tic-tac-toe grid."
    line(-67, 200, -67, -200)
    line(67, 200, 67, -200)
    line(-200, -67, 200, -67)
    line(-200, 67, 200, 67)


def drawx(x, y):
    "Draw X player."
    line(x, y, x + 133, y + 133)
    line(x, y + 133, x + 133, y)


def drawo(x, y):
    "Draw O player."
    up()
    goto(x + 67, y + 5)
    down()
    circle(62)


def floor(value):
    "Round value down to grid with square size 133."
    return ((value + 200) // 133) * 133 - 200


state = {'player': 0}
players = [drawx, drawo]


def tap(x, y):
    "Draw X or O in tapped square."
    x = floor(x)
    y = floor(y)
    player = state['player']
    draw = players[player]
    draw(x, y)
    update()
    state['player'] = not player


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
grid()
update()
onscreenclick(tap)
done()
```

Output :



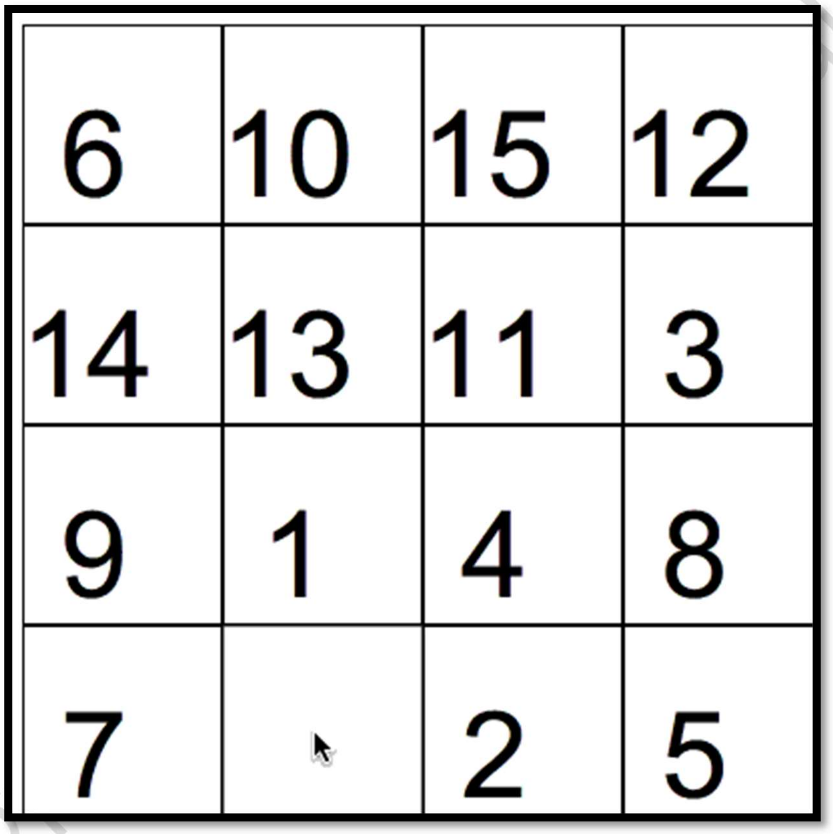
11) Tiles

puzzle game of sliding numbers into place. Click a tile adjacent to the empty square to swap positions. Can you make the tiles count one to fifteen from left to right and bottom to top?

Source Code:

```
from random import *
from turtle import *

from freegames import floor, vector

tiles = {}
neighbors = [
    vector(100, 0),
    vector(-100, 0),
    vector(0, 100),
    vector(0, -100),
]


def load():
    "Load tiles and scramble."
    count = 1

    for y in range(-200, 200, 100):
        for x in range(-200, 200, 100):
            mark = vector(x, y)
            tiles[mark] = count
            count += 1

    tiles[mark] = None

    for count in range(1000):
        neighbor = choice(neighbors)
        spot = mark + neighbor

        if spot in tiles:
            number = tiles[spot]
            tiles[spot] = None
            tiles[mark] = number
            mark = spot


def square(mark, number):
    "Draw white square with black outline and number."
    up()
    goto(mark.x, mark.y)
    down()

    color('black', 'white')
    begin_fill()
    for count in range(4):
        forward(99)
        left(90)
    end_fill()

    if number is None:
        return
    elif number < 10:
        forward(20)

    write(number, font=('Arial', 60, 'normal'))


def tap(x, y):
    "Swap tile and empty square."
    x = floor(x, 100)
    y = floor(y, 100)
    mark = vector(x, y)

    for neighbor in neighbors:
        spot = mark + neighbor

        if spot in tiles and tiles[spot] is None:
            number = tiles[mark]
            tiles[spot] = number
            square(spot, number)
            tiles[mark] = None
            square(mark, None)


def draw():
    "Draw all tiles."
    for mark in tiles:
        square(mark, tiles[mark])
    update()


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
load()
draw()
onscreenclick(tap)
done()
```

Output:



12) Tron

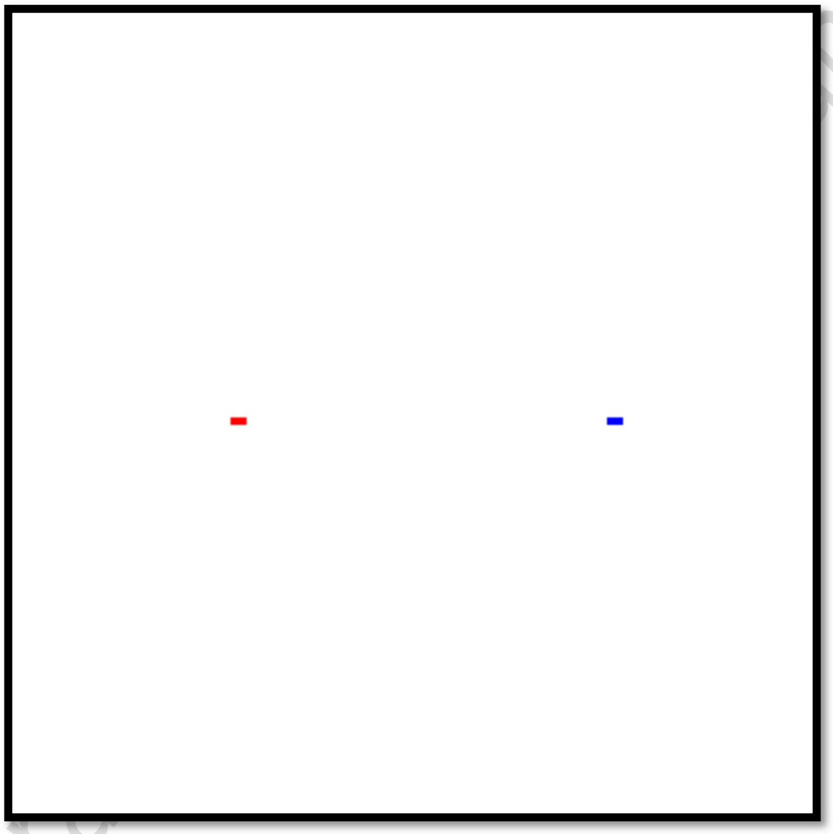
classic arcade game. Use the keyboard to change the direction of your Tron player. Avoid touching the line drawn by your opponent.

Source Code:

```python
from turtle import *

from freegames import square, vector

p1xy = vector(-100, 0)
p1aim = vector(4, 0)
p1body = set()

p2xy = vector(100, 0)
p2aim = vector(-4, 0)
p2body = set()


def inside(head):
    "Return True if head inside screen."
    return -200 < head.x < 200 and -200 < head.y < 200


def draw():
    "Advance players and draw game."
    p1xy.move(p1aim)
    p1head = p1xy.copy()

    p2xy.move(p2aim)
    p2head = p2xy.copy()

    if not inside(p1head) or p1head in p2body:
        print('Player blue wins!')
        return

    if not inside(p2head) or p2head in p1body:
        print('Player red wins!')
        return

    p1body.add(p1head)
    p2body.add(p2head)

    square(p1xy.x, p1xy.y, 3, 'red')
    square(p2xy.x, p2xy.y, 3, 'blue')
    update()
    ontimer(draw, 50)


setup(420, 420, 370, 0)
```

```
hideturtle()
tracer(False)
listen()
onkey(lambda: p1aim.rotate(90), 'a')
onkey(lambda: p1aim.rotate(-90), 'd')
onkey(lambda: p2aim.rotate(90), 'j')
onkey(lambda: p2aim.rotate(-90), 'l')
draw()
done()
```

Output:



13) Life

Conway’s Game of Life. The classic, zero-player, cellular automation created in 1970 by John Conway

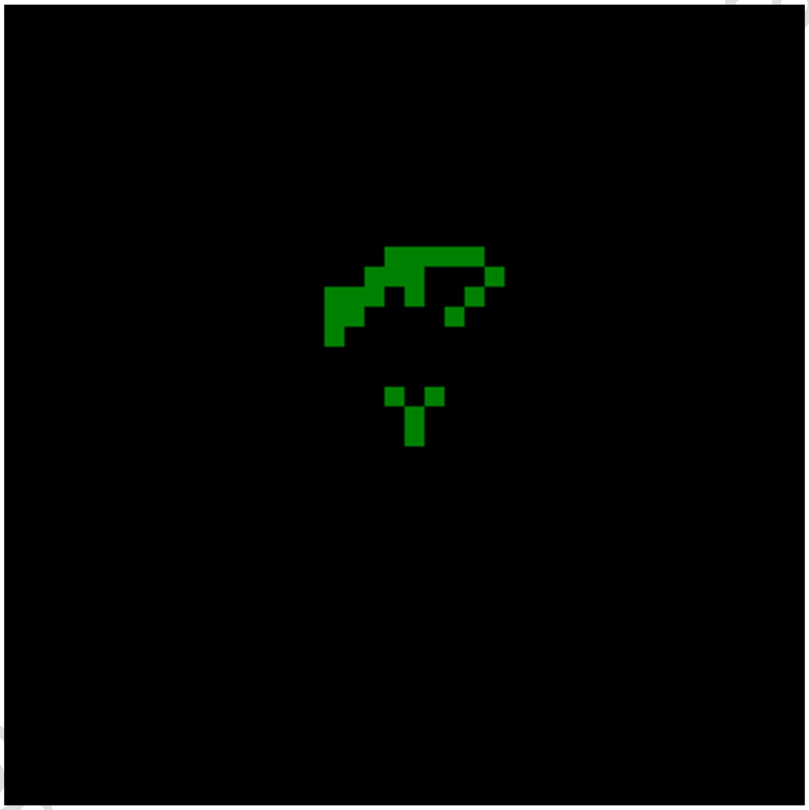
Source code :

```
from random import choice
from turtle import *
```

```python
from freegames import square

cells = {}


def initialize():
    "Randomly initialize the cells."
    for x in range(-200, 200, 10):
        for y in range(-200, 200, 10):
            cells[x, y] = False

    for x in range(-50, 50, 10):
        for y in range(-50, 50, 10):
            cells[x, y] = choice([True, False])


def step():
    "Compute one step in the Game of Life."
    neighbors = {}

    for x in range(-190, 190, 10):
        for y in range(-190, 190, 10):
            count = -cells[x, y]
            for h in [-10, 0, 10]:
                for v in [-10, 0, 10]:
                    count += cells[x + h, y + v]
            neighbors[x, y] = count

    for cell, count in neighbors.items():
        if cells[cell]:
            if count < 2 or count > 3:
                cells[cell] = False
        elif count == 3:
            cells[cell] = True


def draw():
    "Draw all the squares."
    step()
    clear()
    for (x, y), alive in cells.items():
        color = 'green' if alive else 'black'
        square(x, y, 10, color)
    update()
    ontimer(draw, 100)
```

```
setup(420, 420, 370, 0)
hideturtle()
tracer(False)
initialize()
draw()
done()
```

Output:



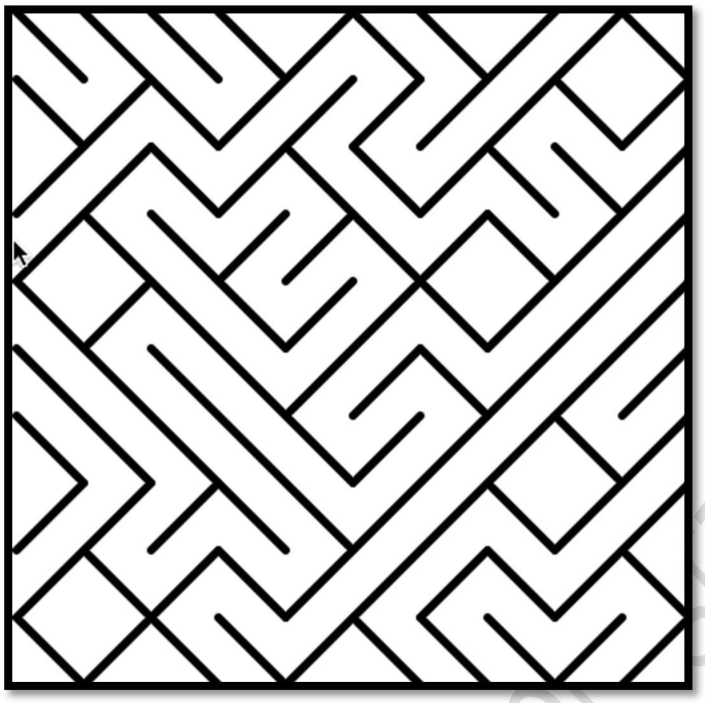
14) Maze

move from one side to another. Inspired by A Universe in One Line of Code with 10 PRINT. Tap the screen to trace a path from one side to another.

Source code :

```
from random import random
from turtle import *

from freegames import line


def draw():
    "Draw maze."
    color('black')
    width(5)

    for x in range(-200, 200, 40):
        for y in range(-200, 200, 40):
            if random() > 0.5:
                line(x, y, x + 40, y + 40)
            else:
                line(x, y + 40, x + 40, y)

    update()


def tap(x, y):
    "Draw line and dot for screen tap."
    if abs(x) > 198 or abs(y) > 198:
        up()
    else:
        down()

    width(2)
    color('red')
    goto(x, y)
    dot(4)


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
draw()
onscreenclick(tap)
done()
```

Output :



15) Fidget

fidget spinner inspired animation. Click the screen to accelerate the fidget spinner.

Source Code

```
from turtle import *

state = {'turn': 0}


def spinner():
    "Draw fidget spinner."
    clear()
    angle = state['turn'] / 10
    right(angle)
    forward(100)
    dot(120, 'red')
    back(100)
    right(120)
    forward(100)
    dot(120, 'green')
    back(100)
    right(120)
    forward(100)
    dot(120, 'blue')
    back(100)
    right(120)
    update()


def animate():
    "Animate fidget spinner."
    if state['turn'] > 0:
        state['turn'] -= 1

    spinner()
    ontimer(animate, 20)


def flick():
    "Flick fidget spinner."
    state['turn'] += 10


setup(420, 420, 370, 0)
hideturtle()
tracer(False)
width(20)
onkey(flick, 'space')
listen()
animate()
done()
```

Output :

THANK YOU