**A\* Chase with Pathfinder algorithm**

Table of Contents

[**Introduction:** 3](#_Toc166409796)

[**Game Description:** 3](#_Toc166409797)

[**Algorithms Used:** 4](#_Toc166409798)

[**Gameplay:** 4](#_Toc166409799)

[**Conclusion:** 5](#_Toc166409800)

[References 5](#_Toc166409801)

[**Coding** 6](#_Toc166409802)

# **Introduction:**

The AI driven game A\* Chase is created within Python. This is a simple however engaging game that demonstrates the application of artificial intelligence within pathfinding. The unique part of the game is that user can move the object and the artificial intelligence-based algorithm automatically detects the path of the object. In this game a user will find a challenge of avoiding anime character while strategically navigating the game to avoid the capture. In order to move the character, the user can use the keys up down left and right.

# **Game Description:**

"A\* Chase" is a 2D game which means the object will have X&Y axis associated with them. Players will control the characters by using the arrow keys. The main task within the game is to avoid the enemy characters capturing the object. For this purpose, the user can move the object using the arrow keys so that the enemy object can be avoided. The game presents the grid-based layout where both the player and the enemy can move horizontally as well as vertically all the four arrow keys are operational in this case(decisionspace, 2023). The objective is for the player to avoid being caught by the enemy and reach out to the safe zone. Once the person reaches out to the safe zone, a message will be displayed regarding user win and the game will be exited. The safe zone will be indicated with a green rectangle.

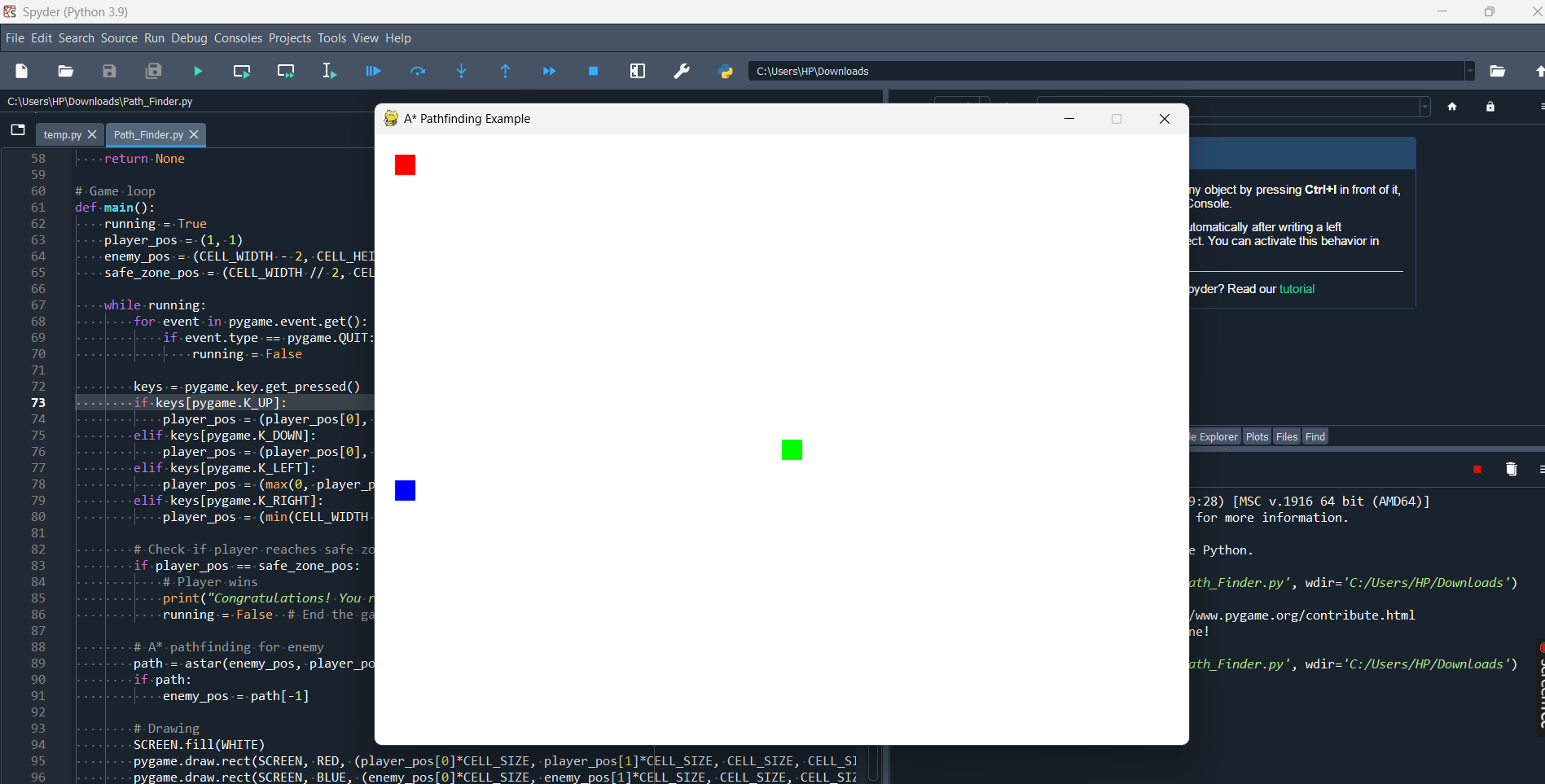


Figure 1: Game UI

The user can move the red object and the AI driven object is of blue colour. Furthermore, the target for the user is to reach out to the green object in order to be safe. In case the user does not reach the safe zone then as well the game terminates by displaying the message you lost(Filip Jerga, 2021).

A computer screen shot of a computer code

Description automatically generated

Figure 2: Safe Zone

Once the user reaches the save zone a message will be displayed that you reached the safe zone and the game will be exited.

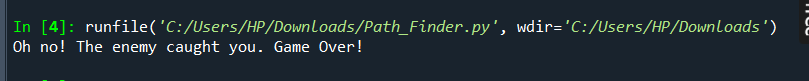


Figure 3: Caught and loss

This message indicates that the red object is caught by the enemy object before reaching out to the safe zone.

# **Algorithms Used:**

The main algorithm that is used is AI driven "A\* Chase" is the A\* pathfinding algorithm. A\* is a popular along with efficient algorithm that is used within games to find the shortest path between the two points. The shortest path in this case will be located by determining the minimum possible cost. The cost in this case will be in terms of distance. The algorithm will calculate the shortest path dynamically between the enemy character and the user character(raddit, 2023). The intelligent pursued behavior will be applied in this case. Basic collision detection mechanisms along with movement logic is also applied. The movement can be controlled by using arrow keys.

# **Gameplay:**

The game play corresponding to this game is described as follows

* First of all the game is required to be started by clicking on the play button. When the game is in play the player character can be controlled by using arrow keys. The motion of the character is not restricted, rather the object can move both horizontally as well as vertically using arrow keys.
* The enemy character thematically tracks the player character using A\* pathfinding algorithm. This algorithm automatically tracks the movement of the player and adjust the position automatically.
* The environment of the game present obstacles to the user. User is required to find out the path toward the safe zone in this case. Once they reach the safe zone the game will be closed and the user is the winner.
* The main objective for the player is to find out the best possible path corresponding to the safe zone in case the safe zone is not reached and the enemy code the user object then the player will lose the game.

# **Conclusion:**

A\* chase game provides the player a challenging gaming experience. The game is started by clicking on the orange button closed up after running the game the object within the game will start moving. When the object is moving the user is required to find out the best possible path toward the safe zone. In case the enemy object which is automatic and detecting the user's path reaches out to the users’ object first then the user will lost the game. In case the user reaches the safe zone then user will win the game. In this game both strategical movements along with quick thinking will be required for the player to reach out to the safe zone. This game serves as a compelling example describing the application of artificial intelligence that can be used in order to enhance gameplay. Furthermore, artificial intelligence can also be used in order to provide the engaging player experience.

# References

decisionspace. (2023). *The Three Types of Objectives in Games — Decision Space*. Web. https://www.decisionspacepodcast.com/articles/the-three-types-of-objectives-in-games

Filip Jerga. (2021). *Unity Fundamentals — Moving a game object | by Filip Jerga | Eincode | Medium*. Web. https://medium.com/eincode/unity-fundamentals-moving-a-game-object-179f708c5d36

raddit. (2023). *[::] Bot with PathFinding AI that autonomeously switches between three Algorithms: Dijkstra’s Concurrent, semi-A\* and a simple Chase algorithm, all compatible with 3d terrain, collision detection & easily configurable search radius - More info in comments : r/Minecraft*. Web. https://www.reddit.com/r/Minecraft/comments/37yiye/bot\_with\_pathfinding\_ai\_that\_autonomeously/

# **Coding**

import pygame

import heapq

# Initialize Pygame

pygame.init()

# Screen dimensions

WIDTH, HEIGHT = 800, 600

SCREEN = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("A\* Pathfinding Example")

# Colors

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLUE = (0, 0, 255)

SAFETY\_COLOR = GREEN # Color representing the safe zone

# Cell dimensions

CELL\_SIZE = 20

CELL\_WIDTH = WIDTH // CELL\_SIZE

CELL\_HEIGHT = HEIGHT // CELL\_SIZE

# Define grid

grid = [[0 for \_ in range(CELL\_WIDTH)] for \_ in range(CELL\_HEIGHT)]

# A\* Pathfinding

def heuristic(a, b):

return abs(a[0] - b[0]) + abs(a[1] - b[1])

def astar(start, end):

open\_set = []

heapq.heappush(open\_set, (0, start))

came\_from = {}

g\_score = {start: 0}

while open\_set:

current = heapq.heappop(open\_set)[1]

if current == end:

path = []

while current in came\_from:

path.append(current)

current = came\_from[current]

return path

for dx, dy in [(0, 1), (1, 0), (0, -1), (-1, 0)]:

next\_cell = (current[0] + dx, current[1] + dy)

if 0 <= next\_cell[0] < CELL\_WIDTH and 0 <= next\_cell[1] < CELL\_HEIGHT:

new\_g = g\_score[current] + 1

if new\_g < g\_score.get(next\_cell, float('inf')):

g\_score[next\_cell] = new\_g

priority = new\_g + heuristic(end, next\_cell)

heapq.heappush(open\_set, (priority, next\_cell))

came\_from[next\_cell] = current

return None

# Game loop

def main():

running = True

player\_pos = (1, 1)

enemy\_pos = (CELL\_WIDTH - 2, CELL\_HEIGHT - 2)

safe\_zone\_pos = (CELL\_WIDTH // 2, CELL\_HEIGHT // 2) # Position of the safe zone

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

keys = pygame.key.get\_pressed()

if keys[pygame.K\_UP]:

player\_pos = (player\_pos[0], max(0, player\_pos[1] - 1))

elif keys[pygame.K\_DOWN]:

player\_pos = (player\_pos[0], min(CELL\_HEIGHT - 1, player\_pos[1] + 1))

elif keys[pygame.K\_LEFT]:

player\_pos = (max(0, player\_pos[0] - 1), player\_pos[1])

elif keys[pygame.K\_RIGHT]:

player\_pos = (min(CELL\_WIDTH - 1, player\_pos[0] + 1), player\_pos[1])

# Check if player reaches safe zone

if player\_pos == safe\_zone\_pos:

# Player wins

print("Congratulations! You reached the safe zone!")

running = False # End the game

# Check if enemy catches the player

if player\_pos == enemy\_pos:

# Player loses

print("Oh no! The enemy caught you. Game Over!")

running = False # End the game

# A\* pathfinding for enemy

path = astar(enemy\_pos, player\_pos)

if path:

enemy\_pos = path[-1]

# Drawing

SCREEN.fill(WHITE)

pygame.draw.rect(SCREEN, RED, (player\_pos[0]\*CELL\_SIZE, player\_pos[1]\*CELL\_SIZE, CELL\_SIZE, CELL\_SIZE))

pygame.draw.rect(SCREEN, BLUE, (enemy\_pos[0]\*CELL\_SIZE, enemy\_pos[1]\*CELL\_SIZE, CELL\_SIZE, CELL\_SIZE))

pygame.draw.rect(SCREEN, SAFETY\_COLOR, (safe\_zone\_pos[0]\*CELL\_SIZE, safe\_zone\_pos[1]\*CELL\_SIZE, CELL\_SIZE, CELL\_SIZE))

pygame.display.flip()

pygame.time.delay(100)

pygame.quit()

if \_\_name\_\_ == "\_\_main\_\_":

main()