1. assets.py

# --- START OF UPGRADED FILE assets.py ---

import pygame

from typing import Dict

from pathlib import Path

from config import GameConfig

class AssetManager:

"""Manages loading and scaling of all game assets."""

def \_\_init\_\_(self, config: GameConfig, theme: Dict, assets\_path: Path):

pygame.mixer.init()

self.config = config; self.theme = theme; self.assets\_path = assets\_path

self.font\_title = pygame.font.SysFont(config.FONT\_NAME, 96, bold=True)

self.font\_large = pygame.font.SysFont(config.FONT\_NAME, 48, bold=True)

self.font\_medium = pygame.font.SysFont(config.FONT\_NAME, 28, bold=True)

self.font\_small = pygame.font.SysFont(config.FONT\_NAME, 18)

self.original\_images = self.\_load\_original\_images()

self.images = self.scale\_images()

self.sounds = self.\_load\_sounds()

# NEW: Load background music

try:

pygame.mixer.music.load(assets\_path / "menu\_music.wav")

print("OK: Loaded background music 'menu\_music.wav'")

except pygame.error as e:

print(f"!! WARNING: Could not load background music: {e}")

def scale\_images(self) -> Dict[str, pygame.Surface]:

scaled = {}; tile\_size = (self.config.TILE\_SIZE, self.config.TILE\_SIZE)

for name, surf in self.original\_images.items():

if name == 'background': scaled[name] = pygame.transform.scale(surf, (self.config.WINDOW\_WIDTH, self.config.WINDOW\_HEIGHT))

elif name == 'logo': scaled[name] = surf # Don't scale logo initially

else: scaled[name] = pygame.transform.scale(surf, tile\_size)

return scaled

def \_load\_original\_images(self) -> Dict[str, pygame.Surface]:

images = {}

image\_files = {

'player\_front':'player\_front.png','player\_back':'player\_back.png','player\_left':'player\_left.png',

'player\_right':'player\_right.png','player\_face':'playerFace.png','wall':'block.png',

'box':'box.png','floor':'ground.png','background':'ground.png','target':'target1.png',

'logo': 'logo.png' # NEW: Logo file

}

print("--- Loading Images ---")

for name, filename in image\_files.items():

path = self.assets\_path / filename

try:

if name == 'player\_front': images['player'] = pygame.image.load(path).convert\_alpha()

images[name] = pygame.image.load(path).convert\_alpha()

print(f"OK: Loaded image '{filename}' as '{name}'")

except pygame.error as e:

print(f"!! FATAL ERROR: Could not load image '{path}': {e}"); raise SystemExit(e)

images['box\_on\_target'] = images['box']

print("----------------------")

return images

def \_load\_sounds(self) -> Dict[str, pygame.mixer.Sound]:

sounds = {}

sound\_files = ["button.wav","move.wav","place\_box.wav","undo.wav","win.wav", "new\_highscore.wav"] # NEW: Sound for ranking up

print("--- Loading Sounds ---")

for f in sound\_files:

name = f.split('.')[0]

try:

sounds[name] = pygame.mixer.Sound(str(self.assets\_path / f))

print(f"OK: Loaded sound '{f}' as sounds['{name}']")

except pygame.error:

print(f"!! WARNING: Could not load sound '{f}'."); sounds[name] = type('DummySound',(),{'play':lambda:None})()

print("----------------------")

return sounds

2. game.py

# --- START OF UPGRADED FILE game.py ---

from typing import Dict, Tuple, List, Optional

import numpy as np

import time

from pathlib import Path

import json

from datetime import datetime

from constants import GameObject

from save\_load import save\_data, load\_data

from assets import AssetManager

from solver import bfs\_solver

from core import get\_initial\_board, get\_targets\_mask, is\_win, move

INITIAL\_LEVELS = [ [[0,0,-1,-1,-1,0],[0,0,-1,1,-1,0],[0,0,-1,0,-1,-1],[-1,-1,-1,2,0,-1],[-1,1,0,2,3,-1],[-1,-1,-1,-1,-1,-1]],[[0,0,0,-1,-1,-1,0,0],[0,-1,-1,-1,1,-1,0,0],[-1,-1,0,0,0,-1,0,0],[-1,0,2,-1,3,-1,-1,-1],[-1,0,0,0,0,0,0,-1],[-1,-1,-1,-1,0,0,0,-1],[0,0,0,-1,-1,-1,-1,-1]],[[0,0,0,-1,-1,-1,0,0],[0,0,-1,-1,1,-1,-1,-1],[0,-1,-1,0,0,0,0,-1],[-1,-1,0,3,0,0,0,-1],[-1,0,2,0,-1,0,0,-1],[-1,0,0,0,0,0,-1,-1],[-1,-1,-1,-1,-1,-1,-1,0]],[[-1,-1,-1,-1,-1,-1,-1,-1],[-1,1,0,0,0,0,0,-1],[-1,0,2,0,2,0,0,-1],[-1,0,2,0,0,0,0,-1],[-1,-1,-1,-1,0,3,-1,-1],[0,0,0,-1,-1,-1,-1,-1]],[[0,0,0,0,0,-1,-1,-1],[0,0,0,0,0,-1,1,-1],[0,-1,-1,-1,-1,-1,0,-1],[0,-1,0,0,0,0,0,-1],[-1,-1,2,0,0,0,0,-1],[-1,0,0,0,-1,-1,-1,-1],[-1,0,0,0,0,3,-1,0],[-1,-1,-1,-1,-1,-1,-1,0]],[[0,0,0,0,0,-1,-1,-1],[0,0,0,0,-1,-1,1,-1],[0,0,0,0,-1,0,0,-1],[0,0,0,0,-1,2,0,-1],[-1,-1,-1,-1,-1,0,0,-1],[-1,0,0,0,0,0,0,-1],[-1,0,0,3,-1,0,0,-1],[-1,-1,-1,-1,-1,-1,-1,-1]],[[0,0,0,0,0,-1,-1,-1,-1],[-1,-1,-1,-1,-1,-1,0,3,-1],[-1,0,2,0,0,0,0,0,-1],[-1,1,0,0,0,-1,0,-1,-1],[-1,-1,-1,0,0,0,0,-1,-1],[0,0,-1,0,0,0,0,0,-1],[0,0,-1,-1,-1,0,-1,0,-1],[0,0,0,0,-1,0,0,0,-1],[0,0,0,0,-1,-1,-1,-1,-1]],[[-1,-1,-1,-1,-1,0],[-1,1,0,0,-1,0],[-1,-1,1,0,-1,-1],[-1,-1,0,2,0,-1],[-1,0,2,-1,0,-1],[-1,0,0,0,3,-1],[-1,-1,-1,-1,-1,-1]],[[-1,-1,-1,-1,-1,0],[-1,1,0,0,-1,0],[-1,0,1,2,-1,-1],[-1,3,2,0,0,-1],[-1,-1,0,0,0,-1],[0,-1,-1,-1,-1,-1]],[[-1,-1,-1,-1,-1,0],[-1,1,0,0,-1,0],[-1,0,1,2,-1,-1],[-1,3,2,0,0,-1],[-1,-1,0,0,0,-1],[0,-1,-1,-1,-1,-1]],[[-1,-1,-1,-1,-1,-1,0,0],[-1,1,-1,0,1,-1,-1,0],[-1,0,0,0,0,0,-1,0],[-1,0,2,0,0,2,-1,-1],[-1,-1,-1,3,-1,0,0,-1],[0,0,-1,0,0,0,0,-1],[0,0,-1,0,0,-1,-1,-1],[0,0,-1,-1,-1,0,0,0]],[[0,0,0,0,0,-1,-1,-1],[-1,-1,-1,-1,-1,-1,1,-1],[-1,0,0,0,2,1,0,-1],[-1,0,0,2,-1,0,3,-1],[-1,-1,-1,0,-1,-1,0,-1],[0,0,-1,0,0,0,0,-1],[0,0,-1,-1,-1,-1,-1,-1]],[[-1,-1,-1,-1,-1,-1,0,0],[-1,1,0,1,0,-1,-1,-1],[-1,0,0,0,2,0,0,-1],[-1,0,-1,0,2,0,0,-1],[-1,0,-1,-1,-1,0,0,-1],[-1,-1,-1,0,-1,-1,-1,-1]],[[-1,-1,-1,-1,-1,-1,0,0],[-1,1,0,0,1,-1,-1,0],[-1,0,-1,2,0,2,-1,0],[-1,0,0,2,3,2,-1,-1],[-1,-1,-1,-1,0,2,0,-1],[0,0,0,-1,0,0,0,-1],[0,0,0,-1,-1,-1,-1,-1]],[[-1,-1,-1,-1,-1,-1,-1,-1],[-1,0,0,-1,1,0,0,-1],[-1,3,2,1,0,0,0,-1],[-1,0,0,2,-1,0,-1,-1],[-1,-1,-1,0,0,0,-1,0],[0,0,-1,-1,-1,-1,-1,0]]]

BASE\_LEVELS: List[dict] = []

CUSTOM\_LEVELS: List[dict] = []

ALL\_SOLUTIONS: Dict[str, Optional[list]] = {}

LEVEL\_STARS: Dict[str, Dict[str, int]] = {}

CURRENT\_PLAYER\_NAME = "Player"

CUSTOM\_LEVELS\_DIR = Path("custom\_levels")

SAVE\_FILE = "sokoban\_save.json"

class GameState:

def \_\_init\_\_(self, level\_key: str, level\_data: list, solution: Optional[List[Tuple[int, int]]], assets: AssetManager):

self.assets = assets; self.level\_key = level\_key; self.solution = solution

self.initial\_board = get\_initial\_board(level\_data); self.target\_mask = get\_targets\_mask(self.initial\_board)

self.move\_stack: List[np.ndarray] = [self.initial\_board.copy()]; self.redo\_stack: List[np.ndarray] = []

self.start\_time = time.time(); self.win\_time: Optional[float] = None; self.is\_won = False

self.auto\_play = False; self.auto\_play\_idx = 0; self.auto\_play\_speed = 0.1

self.player\_direction: Tuple[int, int] = (1, 0)

@property

def current\_board(self) -> np.ndarray: return self.move\_stack[-1]

def perform\_move(self, direction: Tuple[int, int]) -> bool:

if self.is\_won or self.auto\_play: return False

new\_board = move(self.current\_board, direction, self.target\_mask)

if new\_board is not None:

self.player\_direction = direction

if np.count\_nonzero((self.current\_board==GameObject.BOX.value)&self.target\_mask) < np.count\_nonzero((new\_board==GameObject.BOX.value)&self.target\_mask):

self.assets.sounds['place\_box'].play()

else: self.assets.sounds['move'].play()

self.move\_stack.append(new\_board); self.redo\_stack.clear(); self.check\_win(); return True

return False

def undo(self):

if len(self.move\_stack) > 1 and not self.auto\_play: self.assets.sounds['undo'].play(); self.redo\_stack.append(self.move\_stack.pop())

def restart(self):

self.move\_stack=[self.initial\_board.copy()]; self.redo\_stack=[]; self.start\_time=time.time()

self.is\_won=False; self.win\_time=None; self.auto\_play=False; self.player\_direction=(1,0)

def start\_solver(self):

if self.solution: self.restart(); self.auto\_play=True; self.auto\_play\_idx=0

def step\_solver(self):

if self.auto\_play and self.auto\_play\_idx < len(self.solution):

direction = self.solution[self.auto\_play\_idx]; self.player\_direction=direction

new\_board = move(self.current\_board, direction, self.target\_mask)

if new\_board is not None: self.move\_stack.append(new\_board)

self.auto\_play\_idx += 1; self.check\_win()

else: self.auto\_play=False

def check\_win(self):

global LEVEL\_STARS

if not self.is\_won and is\_win(self.current\_board, self.target\_mask):

self.is\_won = True; self.win\_time = time.time(); self.auto\_play=False; self.assets.sounds['win'].play()

moves = len(self.move\_stack)-1; optimal\_moves = len(self.solution) if self.solution else float('inf')

stars = 1

if moves <= optimal\_moves \* 1.5: stars=2

if moves <= optimal\_moves: stars=3

player\_stars = LEVEL\_STARS.setdefault(CURRENT\_PLAYER\_NAME, {})

player\_stars[self.level\_key] = max(player\_stars.get(self.level\_key, 0), stars)

save\_data(LEVEL\_STARS, Path(SAVE\_FILE))

def get\_stars\_for\_player(player\_name: str) -> Dict[str, int]: return LEVEL\_STARS.get(player\_name, {})

def get\_player\_rankings() -> List[dict]:

scores = []

for player\_name, stars\_dict in LEVEL\_STARS.items():

total\_stars = sum(stars\_dict.values())

scores.append({"name": player\_name, "stars": total\_stars})

scores.sort(key=lambda s: s["stars"], reverse=True)

return scores

def initialize\_game\_data():

global BASE\_LEVELS, CUSTOM\_LEVELS, ALL\_SOLUTIONS, LEVEL\_STARS, CURRENT\_PLAYER\_NAME

BASE\_LEVELS = [{'key': f'base\_{i}', 'name': f'Level {i+1}', 'data': lvl} for i, lvl in enumerate(INITIAL\_LEVELS)]

CUSTOM\_LEVELS\_DIR.mkdir(exist\_ok=True)

CUSTOM\_LEVELS = []

for file\_path in CUSTOM\_LEVELS\_DIR.glob("\*.json"):

try:

level\_data = load\_data(file\_path)

player\_name = file\_path.stem.split('\_')[0]

CUSTOM\_LEVELS.append({'key': file\_path.name, 'name': f"By {player\_name}", 'data': level\_data})

except Exception as e: print(f"Error loading custom level {file\_path}: {e}")

saved\_progress = load\_data(Path(SAVE\_FILE))

if saved\_progress and isinstance(saved\_progress, dict):

LEVEL\_STARS = saved\_progress

player\_list = list(LEVEL\_STARS.keys())

if player\_list: CURRENT\_PLAYER\_NAME = player\_list[-1]

else: LEVEL\_STARS = {}

print("Pre-computing solutions..."); ALL\_SOLUTIONS.clear()

for level in BASE\_LEVELS + CUSTOM\_LEVELS:

board=get\_initial\_board(level['data']); solution=bfs\_solver(board)

ALL\_SOLUTIONS[level['key']] = solution

status="SOLVABLE"if solution else"UNSOLVABLE"; print(f" - {level['key']}: {status}(len:{len(solution)if solution else'N/A'})")

print("Done.")

def save\_custom\_level(level\_data: list):

filename = f"{CURRENT\_PLAYER\_NAME}\_{datetime.now().strftime('%Y-%m-%d\_%H%M%S')}.json"

save\_path = CUSTOM\_LEVELS\_DIR / filename

save\_data(level\_data, save\_path)

print(f"New custom level saved to {save\_path}")

initialize\_game\_data()

def solve\_new\_level(level\_data: list):

print("Checking level solvability...");board=get\_initial\_board(level\_data)

solution = bfs\_solver(board)

status="SOLVABLE"if solution else"UNSOLVABLE";print(f"Level Check:{status}(len:{len(solution)if solution else'N/A'})")

return solution is not None

3. main.py

# --- START OF UPGRADED FILE main.py ---

import asyncio

import pygame

from pathlib import Path

import platform

from config import GameConfig, DEFAULT\_THEME

from assets import AssetManager

import game

from ui import show\_splash\_screen, show\_main\_menu, show\_player\_select\_screen, show\_mode\_select, show\_level\_select, play\_level, level\_editor, show\_high\_scores

async def main():

"""Main entry point for the Sokoban game."""

pygame.init()

config = GameConfig()

screen = pygame.display.set\_mode((config.WINDOW\_WIDTH, config.WINDOW\_HEIGHT), pygame.RESIZABLE)

pygame.display.set\_caption("Sokoban by Redha & Rooney")

assets\_path = Path(\_\_file\_\_).parent / "assets"

assets = AssetManager(config, DEFAULT\_THEME, assets\_path)

game.initialize\_game\_data()

# Start menu music

if pygame.mixer.get\_init() and pygame.mixer.music.get\_volume() > 0:

pygame.mixer.music.play(-1, 0.0, 5000)

game\_state = "SPLASH"

while game\_state != "QUIT":

if game\_state == "SPLASH": game\_state = await show\_splash\_screen(screen, assets)

elif game\_state == "MAIN\_MENU": game\_state = await show\_main\_menu(screen, assets)

elif game\_state == "PLAYER\_SELECT": game\_state = await show\_player\_select\_screen(screen, assets)

elif game\_state == "HIGH\_SCORES": game\_state = await show\_high\_scores(screen, assets)

elif game\_state == "MODE\_SELECT":

mode = await show\_mode\_select(screen, assets)

if mode in ["PLAYER\_SELECT", "MAIN\_MENU"]: game\_state = mode; continue

levels\_to\_show = game.BASE\_LEVELS if mode == "main" else game.CUSTOM\_LEVELS

next\_state, level\_idx, level\_key = await show\_level\_select(screen, assets, levels\_to\_show, mode)

if next\_state == "PLAY":

from ui import current\_level\_idx, current\_level\_key, current\_level\_mode

current\_level\_idx, current\_level\_key, current\_level\_mode = level\_idx, level\_key, mode

game\_state = next\_state

elif game\_state == "EDITOR":

game\_state = await level\_editor(screen, assets)

elif game\_state == "PLAY":

game\_state = await play\_level(screen, assets)

if game\_state != "QUIT":

pygame.mixer.music.play(-1, 0.0, 2000)

for event in pygame.event.get(pygame.VIDEORESIZE):

screen = pygame.display.set\_mode((event.w, event.h), pygame.RESIZABLE)

config.TILE\_SIZE = int(64 \* min(event.w / config.WINDOW\_WIDTH, event.h / config.WINDOW\_HEIGHT))

assets.images = assets.scale\_images()

pygame.quit()

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

if platform.system() == "Emscripten": asyncio.ensure\_future(main())

else: asyncio.run(main())

4. ui.py

# --- START OF UPGRADED FILE ui.py ---

import asyncio

import pygame

import numpy as np

import time

from typing import Tuple, Optional, List

from pathlib import Path

import random

from config import GameConfig, DEFAULT\_THEME as THEME

from assets import AssetManager

import game

from core import get\_initial\_board

from save\_load import save\_data, load\_data

current\_level\_idx=0; current\_level\_key=''; current\_level\_mode='main'; player\_animator=None

class AnimatedPlayer:

def \_\_init\_\_(self,initial\_sprite:pygame.Surface):self.original\_sprite=initial\_sprite;self.sprite=initial\_sprite;self.y\_offset=0;self.breath\_timer=0;self.move\_timer=0;self.squash=1.0

def set\_sprite(self,new\_sprite:pygame.Surface):self.original\_sprite=new\_sprite

def update(self):

self.breath\_timer+=0.1;self.y\_offset=np.sin(self.breath\_timer)\*2

if self.move\_timer > 0: self.move\_timer-=1;progress=(10-self.move\_timer)/10;self.squash=1.0+0.3\*np.sin(progress\*np.pi)

else: self.squash=1.0

w,h=self.original\_sprite.get\_size();squash\_h=int(h\*(1/self.squash));squash\_w=int(w\*self.squash);self.sprite=pygame.transform.scale(self.original\_sprite,(squash\_w,squash\_h))

def trigger\_move(self): self.move\_timer=10

class FloatingBox:

def \_\_init\_\_(self, W, H, assets):

self.W, self.H=W,H;self.x=random.randint(0,W);self.y=random.randint(0,H)

self.speed=random.uniform(0.5,1.5);self.size=random.randint(40,80)

self.sprite=pygame.transform.scale(assets.images['box'],(self.size,self.size))

def update(self):

self.y-=self.speed

if self.y<-self.size:self.y=self.H;self.x=random.randint(0,self.W)

def draw\_board\_and\_objects(screen,board,assets,offset=(0,0),is\_editor=False,target\_mask=None,player\_direction=(1,0)):

h,w=board.shape;start\_x,start\_y=offset;bg\_tile=assets.images['floor']

for i in range(h):

for j in range(w):

rect=pygame.Rect(start\_x+j\*assets.config.TILE\_SIZE,start\_y+i\*assets.config.TILE\_SIZE,assets.config.TILE\_SIZE,assets.config.TILE\_SIZE)

cell=board[i,j];is\_target\_location=False

if target\_mask is not None and i<target\_mask.shape[0] and j<target\_mask.shape[1]:is\_target\_location=target\_mask[i,j]

elif is\_editor and cell==game.GameObject.TARGET.value:is\_target\_location=True

screen.blit(bg\_tile,rect.topleft)

if is\_target\_location:screen.blit(assets.images['target'],rect.topleft)

if cell==game.GameObject.WALL.value:screen.blit(assets.images['wall'],rect.topleft)

elif cell==game.GameObject.BOX.value:screen.blit(assets.images['box'],rect.topleft)

elif cell==game.GameObject.PLAYER.value:

sprite\_key='player\_front'

if player\_direction==(-1,0):sprite\_key='player\_back'

elif player\_direction==(0,-1):sprite\_key='player\_left'

elif player\_direction==(0,1):sprite\_key='player\_right'

player\_sprite=assets.images[sprite\_key]

if player\_animator and not is\_editor:player\_animator.set\_sprite(player\_sprite);player\_animator.update();player\_sprite=player\_animator.sprite

player\_rect=player\_sprite.get\_rect(center=rect.center)

if player\_animator and not is\_editor:player\_rect.y+=player\_animator.y\_offset

screen.blit(player\_sprite,player\_rect)

def draw\_gradient\_rect(screen,rect,color1,color2,vertical=True):

surface=pygame.Surface((rect.width,rect.height))

for i in range(rect.height if vertical else rect.width):

alpha=i/(rect.height if vertical else rect.width);color=[int(c1+(c2-c1)\*alpha) for c1,c2 in zip(color1,color2)]

if vertical:pygame.draw.line(surface,color,(0,i),(rect.width,i))

else:pygame.draw.line(surface,color,(i,0),(i,rect.height))

screen.blit(surface,rect.topleft)

def draw\_header\_button(screen,rect,symbol\_text,assets,enabled=True,toggled=False):

theme=assets.theme;shadow\_rect=rect.copy();shadow\_rect.y+=3

pygame.draw.rect(screen,theme['UI\_BTN\_SHADOW'],shadow\_rect,border\_radius=8)

base\_color=theme['UI\_BTN\_HOVER'] if toggled else theme['UI\_BTN']

color=base\_color if enabled else tuple(int(c\*0.8) for c in base\_color)

gradient\_color=tuple(int(c\*0.9) for c in color);draw\_gradient\_rect(screen,rect,color,gradient\_color)

if toggled:pygame.draw.rect(screen,theme['UI\_BTN\_TOGGLE\_ON'],rect,2,border\_radius=8)

text\_surf=assets.font\_medium.render(symbol\_text,True,theme['UI\_BTN\_TEXT']);screen.blit(text\_surf,text\_surf.get\_rect(center=rect.center))

def draw\_star(surface,center,size,color):

points=[];angle=np.pi/2

for \_ in range(5):points.append((center[0]+size\*np.cos(angle),center[1]-size\*np.sin(angle)));angle+=2\*np.pi/5;points.append((center[0]+size/2\*np.cos(angle),center[1]-size/2\*np.sin(angle)));angle+=2\*np.pi/5

pygame.draw.polygon(surface,color,points)

async def fade\_transition(screen,assets,fade\_in=True,duration=0.25):

W,H=screen.get\_size();overlay=pygame.Surface((W,H));clock=pygame.time.Clock()

steps=int(duration\*assets.config.FPS)

for i in range(steps):alpha=(i/steps) if fade\_in else (1-i/steps);overlay.set\_alpha(int(alpha\*255));overlay.fill(assets.theme['BG']);screen.blit(overlay,(0,0));pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

async def show\_splash\_screen(screen,assets):

await fade\_transition(screen,assets,fade\_in=True);W,H=screen.get\_size();clock=pygame.time.Clock();start\_time=time.time()

while time.time()-start\_time<2:

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT"

if event.type==pygame.MOUSEBUTTONDOWN or event.type==pygame.KEYDOWN:return "MAIN\_MENU"

screen.fill(assets.theme['BG']);

if 'logo' in assets.images:

logo\_img = assets.images['logo']; logo\_rect=logo\_img.get\_rect(center=(W//2,H//2-50));screen.blit(logo\_img,logo\_rect)

title\_surf=assets.font\_title.render("SOKOBAN",True,assets.theme['TEXT']);screen.blit(title\_surf,title\_surf.get\_rect(centerx=W//2,centery=H//2+50))

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

return "MAIN\_MENU"

async def show\_main\_menu(screen,assets):

await fade\_transition(screen,assets,fade\_in=True);W,H=screen.get\_size()

buttons={"PLAY":pygame.Rect(W//2-150,H//2-80,300,70),"BUILD NEW BOARD":pygame.Rect(W//2-150,H//2+10,300,70),"HIGH SCORES":pygame.Rect(W//2-150,H//2+100,300,70),"QUIT":pygame.Rect(W//2-150,H//2+190,300,70)}

boxes=[FloatingBox(W,H,assets) for \_ in range(15)];clock=pygame.time.Clock()

while True:

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT"

if event.type==pygame.MOUSEBUTTONDOWN and event.button==1:

pos=event.pos

if buttons["PLAY"].collidepoint(pos):assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "PLAYER\_SELECT"

if buttons["BUILD NEW BOARD"].collidepoint(pos):assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "EDITOR"

if buttons["HIGH SCORES"].collidepoint(pos):assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "HIGH\_SCORES"

if buttons["QUIT"].collidepoint(pos):assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "QUIT"

screen.fill(assets.theme['BG']);

if 'background' in assets.images:screen.blit(assets.images['background'],(0,0))

for box in boxes:box.update();screen.blit(box.sprite,(box.x,box.y))

if 'logo' in assets.images:

logo\_rect=assets.images['logo'].get\_rect(center=(W//2,H//4));screen.blit(assets.images['logo'],logo\_rect)

for name,rect in buttons.items():draw\_header\_button(screen,rect,name,assets,toggled=rect.collidepoint(pygame.mouse.get\_pos()))

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

async def show\_rank\_popup(screen,assets,old\_rank,new\_rank):

if new\_rank < old\_rank and new\_rank <= 3: assets.sounds.get('new\_highscore', lambda:None).play()

else: assets.sounds['button'].play()

W,H=screen.get\_size();overlay=pygame.Surface((W,H),pygame.SRCALPHA);overlay.fill((0,0,0,180));popup\_rect=pygame.Rect(W//2-250,H//2-100,500,200)

clock=pygame.time.Clock();start\_time=time.time()

while time.time()-start\_time<5:

for event in pygame.event.get():

if event.type in (pygame.QUIT,pygame.MOUSEBUTTONDOWN,pygame.KEYDOWN):return

screen.blit(overlay,(0,0));draw\_header\_button(screen,popup\_rect,"",assets)

if new\_rank < old\_rank:msg=f"You are now ranked #{new\_rank}!";color=assets.theme['WIN']

else:msg=f"You are still ranked #{new\_rank}.";color=assets.theme['TEXT']

rank\_surf=assets.font\_large.render("RANKING UPDATE",True,assets.theme['TEXT']);screen.blit(rank\_surf,rank\_surf.get\_rect(centerx=popup\_rect.centerx,y=popup\_rect.top+20))

msg\_surf=assets.font\_medium.render(msg,True,color);screen.blit(msg\_surf,msg\_surf.get\_rect(center=popup\_rect.center))

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

async def show\_high\_scores(screen,assets):

await fade\_transition(screen,assets,fade\_in=True);W,H=screen.get\_size();back\_button=pygame.Rect(W//2-150,H-120,300,50)

scores=game.get\_player\_rankings();clock=pygame.time.Clock()

while True:

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT"

if event.type==pygame.MOUSEBUTTONDOWN and event.button==1 and back\_button.collidepoint(event.pos):

assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "MAIN\_MENU"

screen.fill(assets.theme['BG']);

if 'background' in assets.images:screen.blit(assets.images['background'],(0,0))

if 'logo' in assets.images:logo\_rect=assets.images['logo'].get\_rect(center=(W//2,80));screen.blit(pygame.transform.smoothscale(assets.images['logo'],(100,100)),logo\_rect)

title\_surf=assets.font\_large.render("HIGH SCORES",True,assets.theme['TEXT']);screen.blit(title\_surf,title\_surf.get\_rect(centerx=W/2,y=150))

for i,score\_data in enumerate(scores[:8]):

y\_pos=250+i\*50

rank\_surf=assets.font\_medium.render(f"#{i+1}",True,assets.theme['TEXT']);screen.blit(rank\_surf,rank\_surf.get\_rect(x=W/2-200,centery=y\_pos))

name\_surf=assets.font\_medium.render(score\_data['name'],True,assets.theme['TEXT']);screen.blit(name\_surf,name\_surf.get\_rect(x=W/2-100,centery=y\_pos))

star\_surf=assets.font\_medium.render(f"{score\_data['stars']} ★",True,assets.theme['STAR']);screen.blit(star\_surf,star\_surf.get\_rect(right=W/2+200,centery=y\_pos))

draw\_header\_button(screen,back\_button,"BACK",assets,toggled=back\_button.collidepoint(pygame.mouse.get\_pos()))

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

async def show\_player\_select\_screen(screen,assets):

await fade\_transition(screen,assets,fade\_in=True);W,H=screen.get\_size();players=list(game.LEVEL\_STARS.keys())

input\_box=pygame.Rect(W//2-150,H-200,300,50);back\_button=pygame.Rect(W//2-150,H-120,300,50)

new\_player\_name="";input\_active=False;clock=pygame.time.Clock()

while True:

player\_buttons={name:pygame.Rect(W//2-150,150+i\*70,300,60) for i,name in enumerate(players)}

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT"

if event.type==pygame.MOUSEBUTTONDOWN and event.button==1:

pos=event.pos

if back\_button.collidepoint(pos):assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "MAIN\_MENU"

input\_active=input\_box.collidepoint(pos)

for name,rect in player\_buttons.items():

if rect.collidepoint(pos):game.CURRENT\_PLAYER\_NAME=name;assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "MODE\_SELECT"

if event.type==pygame.KEYDOWN and input\_active:

if event.key==pygame.K\_RETURN:

if new\_player\_name and new\_player\_name not in players:

game.CURRENT\_PLAYER\_NAME=new\_player\_name;game.LEVEL\_STARS[game.CURRENT\_PLAYER\_NAME]={};save\_data(game.LEVEL\_STARS,Path(game.SAVE\_FILE))

assets.sounds['win'].play();await fade\_transition(screen,assets,fade\_in=False);return "MODE\_SELECT"

elif new\_player\_name in players:

game.CURRENT\_PLAYER\_NAME=new\_player\_name;assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "MODE\_SELECT"

elif event.key==pygame.K\_BACKSPACE:new\_player\_name=new\_player\_name[:-1]

else:new\_player\_name+=event.unicode

screen.fill(assets.theme['BG']);

if 'background' in assets.images:screen.blit(assets.images['background'],(0,0))

title\_surf=assets.font\_large.render("SELECT PLAYER",True,assets.theme['TEXT']);screen.blit(title\_surf,title\_surf.get\_rect(centerx=W/2,y=50))

for name,rect in player\_buttons.items():draw\_header\_button(screen,rect,name,assets,toggled=(name==game.CURRENT\_PLAYER\_NAME or rect.collidepoint(pygame.mouse.get\_pos())))

pygame.draw.rect(screen,assets.theme['UI\_BTN\_HOVER'] if input\_active else assets.theme['UI\_BTN'],input\_box,border\_radius=8);pygame.draw.rect(screen,assets.theme['UI\_BTN\_SHADOW'],input\_box,2,border\_radius=8)

input\_surf=assets.font\_medium.render(new\_player\_name or "Enter new name...",True,assets.theme['UI\_BTN\_TEXT']);screen.blit(input\_surf,input\_surf.get\_rect(center=input\_box.center))

draw\_header\_button(screen,back\_button,"BACK",assets,toggled=back\_button.collidepoint(pygame.mouse.get\_pos()))

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

async def show\_mode\_select(screen,assets):

await fade\_transition(screen,assets,fade\_in=True);W,H=screen.get\_size()

buttons={"MAIN STORY":pygame.Rect(W//2-150,H//2-80,300,70),"COMMUNITY LEVELS":pygame.Rect(W//2-150,H//2+10,300,70)}

back\_button=pygame.Rect(20,20,100,50);clock=pygame.time.Clock()

while True:

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT"

if event.type==pygame.MOUSEBUTTONDOWN and event.button==1:

pos=event.pos

if back\_button.collidepoint(pos):assets.sounds['button'].play();return "PLAYER\_SELECT"

if buttons["MAIN STORY"].collidepoint(pos):assets.sounds['button'].play();return "main"

if buttons["COMMUNITY LEVELS"].collidepoint(pos):assets.sounds['button'].play();return "custom"

screen.fill(assets.theme['BG']);

if 'background' in assets.images:screen.blit(assets.images['background'],(0,0))

title\_surf=assets.font\_large.render("SELECT MODE",True,assets.theme['TEXT']);screen.blit(title\_surf,title\_surf.get\_rect(centerx=W/2,y=50))

for name,rect in buttons.items():draw\_header\_button(screen,rect,name,assets,toggled=rect.collidepoint(pygame.mouse.get\_pos()))

draw\_header\_button(screen,back\_button,"BACK",assets,toggled=back\_button.collidepoint(pygame.mouse.get\_pos()))

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

async def show\_level\_select(screen,assets,levels\_to\_display:List[dict],mode:str):

await fade\_transition(screen,assets,fade\_in=True);W,H=screen.get\_size();

back\_btn=pygame.Rect(20,50,100,50);player\_stars=game.get\_stars\_for\_player(game.CURRENT\_PLAYER\_NAME)

clock=pygame.time.Clock();scroll\_y=0;num\_levels=len(levels\_to\_display);cols,btn\_size,spacing=8,120,20

grid\_start\_y,grid\_end\_y=150,H-20;available\_height=grid\_end\_y-grid\_start\_y

num\_rows=(num\_levels+cols-1)//cols if num\_levels>0 else 0

total\_grid\_height=num\_rows\*btn\_size+max(0,num\_rows-1)\*spacing;max\_scroll\_y=max(0,total\_grid\_height-available\_height)

start\_x=(W-(cols\*btn\_size+(cols-1)\*spacing))//2

while True:

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT",-1,None

if event.type==pygame.MOUSEWHEEL:scroll\_y-=event.y\*20;scroll\_y=max(0,min(scroll\_y,max\_scroll\_y))

if event.type==pygame.MOUSEBUTTONDOWN and event.button==1:

pos=event.pos

if back\_btn.collidepoint(pos):assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "MODE\_SELECT",-1,None

for i in range(num\_levels):

rect=pygame.Rect(start\_x+(i%cols)\*(btn\_size+spacing),grid\_start\_y+(i//cols)\*(btn\_size+spacing)-scroll\_y,btn\_size,btn\_size)

if rect.collidepoint(event.pos):

level\_info=levels\_to\_display[i];assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False)

return "PLAY",i,level\_info['key']

screen.fill(assets.theme['BG']);

if 'background' in assets.images:screen.blit(assets.images['background'],(0,0))

title="Main Levels" if mode=="main" else "Community Levels";title\_text=assets.font\_large.render(title,True,assets.theme['TEXT']);screen.blit(title\_text,title\_text.get\_rect(centerx=W//2,y=50))

draw\_header\_button(screen,back\_btn,"BACK",assets,toggled=back\_btn.collidepoint(pygame.mouse.get\_pos()))

grid\_surface=pygame.Surface((W,total\_grid\_height),pygame.SRCALPHA)

for i in range(num\_levels):

level\_info=levels\_to\_display[i];rect=pygame.Rect(start\_x+(i%cols)\*(btn\_size+spacing),(i//cols)\*(btn\_size+spacing),btn\_size,btn\_size)

level\_name=level\_info.get('name',str(i+1));draw\_header\_button(grid\_surface,rect,level\_name,assets,toggled=rect.move(0,grid\_start\_y-scroll\_y).collidepoint(pygame.mouse.get\_pos()))

star\_count=player\_stars.get(level\_info['key'],0)

for s in range(3):star\_center=(rect.centerx-25+s\*25,rect.bottom-20);color=assets.theme['STAR'] if s<star\_count else (80,90,100);draw\_star(grid\_surface,star\_center,10,color)

screen.blit(grid\_surface,(0,grid\_start\_y),(0,scroll\_y,W,available\_height))

if max\_scroll\_y>0:

scrollbar\_bg\_rect=pygame.Rect(W-20,grid\_start\_y,15,available\_height);pygame.draw.rect(screen,assets.theme['UI\_BTN\_SHADOW'],scrollbar\_bg\_rect,border\_radius=7)

handle\_height=max(20,available\_height\*(available\_height/total\_grid\_height));handle\_y=scrollbar\_bg\_rect.y+(scroll\_y/max\_scroll\_y)\*(available\_height-handle\_height)

scrollbar\_handle\_rect=pygame.Rect(W-20,handle\_y,15,handle\_height);draw\_header\_button(screen,scrollbar\_handle\_rect,"",assets)

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

async def play\_level(screen,assets):

global current\_level\_idx,current\_level\_key,current\_level\_mode,player\_animator

pygame.mixer.music.fadeout(1000)

await fade\_transition(screen,assets,fade\_in=True)

levels\_list=game.BASE\_LEVELS if current\_level\_mode=='main' else game.CUSTOM\_LEVELS

level\_data=levels\_list[current\_level\_idx]['data'];solution=game.ALL\_SOLUTIONS.get(current\_level\_key)

game\_state=game.GameState(current\_level\_key,level\_data,solution,assets);W,H=screen.get\_size()

player\_animator=AnimatedPlayer(assets.images['player\_front'])

level\_text\_str=f"Level {current\_level\_idx+1}" if current\_level\_mode=='main' else "Custom Level";level\_text=assets.font\_large.render(level\_text\_str,True,assets.theme['TEXT']);text\_rect=level\_text.get\_rect(centerx=W//2,centery=assets.config.HEADER\_HEIGHT//2);btn\_size\_h,btn\_spacing=60,15;nav\_buttons={"menu":pygame.Rect(btn\_spacing,text\_rect.centery-btn\_size\_h//2,btn\_size\_h,btn\_size\_h),"prev":pygame.Rect(text\_rect.left-btn\_size\_h-btn\_spacing,text\_rect.centery-btn\_size\_h//2,btn\_size\_h,btn\_size\_h),"next":pygame.Rect(text\_rect.right+btn\_spacing,text\_rect.centery-btn\_size\_h//2,btn\_size\_h,btn\_size\_h)};btn\_w\_b,btn\_h\_b,btn\_s\_b=110,40,15;bottom\_buttons={"Restart":pygame.Rect(20,H-btn\_h\_b-20,btn\_w\_b,btn\_h\_b),"Solve":pygame.Rect(20+btn\_w\_b+btn\_s\_b,H-btn\_h\_b-20,btn\_w\_b,btn\_h\_b)};board\_w=game\_state.current\_board.shape[1]\*assets.config.TILE\_SIZE;board\_h=game\_state.current\_board.shape[0]\*assets.config.TILE\_SIZE;board\_offset=((W-board\_w)//2,assets.config.HEADER\_HEIGHT+(H-assets.config.HEADER\_HEIGHT-board\_h)//2);last\_auto\_move=time.time();clock=pygame.time.Clock();move\_flash=0

rankings=game.get\_player\_rankings();old\_rank=next((i+1 for i,p in enumerate(rankings) if p['name']==game.CURRENT\_PLAYER\_NAME),len(rankings)+1)

while True:

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT"

if event.type==pygame.KEYDOWN:

key\_map={pygame.K\_UP:(-1,0),pygame.K\_w:(-1,0),pygame.K\_DOWN:(1,0),pygame.K\_s:(1,0),pygame.K\_LEFT:(0,-1),pygame.K\_a:(0,-1),pygame.K\_RIGHT:(0,1),pygame.K\_d:(0,1)}

if event.key in key\_map:

if game\_state.perform\_move(key\_map[event.key]):player\_animator.trigger\_move();move\_flash=5

elif event.key==pygame.K\_z or event.key==pygame.K\_u:game\_state.undo()

elif event.key in (pygame.K\_ESCAPE,pygame.K\_m):

new\_rank=next((i+1 for i,p in enumerate(game.get\_player\_rankings()) if p['name']==game.CURRENT\_PLAYER\_NAME),old\_rank+1)

await show\_rank\_popup(screen,assets,old\_rank,new\_rank);await fade\_transition(screen,assets,fade\_in=False);return "MODE\_SELECT"

elif event.key==pygame.K\_r:game\_state.restart();assets.sounds['button'].play()

elif event.key==pygame.K\_h and game\_state.solution:game\_state.start\_solver();assets.sounds['button'].play()

if event.type==pygame.MOUSEBUTTONDOWN and event.button==1:

pos=event.pos

if nav\_buttons["menu"].collidepoint(pos):

new\_rank=next((i+1 for i,p in enumerate(game.get\_player\_rankings()) if p['name']==game.CURRENT\_PLAYER\_NAME),old\_rank+1)

await show\_rank\_popup(screen,assets,old\_rank,new\_rank);await fade\_transition(screen,assets,fade\_in=False);return "MODE\_SELECT"

if nav\_buttons["prev"].collidepoint(pos) and current\_level\_idx>0:current\_level\_idx-=1;current\_level\_key=levels\_list[current\_level\_idx]['key'];assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "PLAY"

if nav\_buttons["next"].collidepoint(pos) and current\_level\_idx<len(levels\_list)-1:current\_level\_idx+=1;current\_level\_key=levels\_list[current\_level\_idx]['key'];assets.sounds['button'].play();await fade\_transition(screen,assets,fade\_in=False);return "PLAY"

if bottom\_buttons["Restart"].collidepoint(pos):game\_state.restart();assets.sounds['button'].play()

if bottom\_buttons["Solve"].collidepoint(pos) and game\_state.solution:game\_state.start\_solver();assets.sounds['button'].play()

if game\_state.auto\_play and time.time()-last\_auto\_move>game\_state.auto\_play\_speed:game\_state.step\_solver();player\_animator.trigger\_move();move\_flash=5;last\_auto\_move=time.time()

screen.fill(assets.theme['BG'])

if 'background' in assets.images:screen.blit(assets.images['background'],(0,0))

if move\_flash>0:flash\_surface=pygame.Surface((W,H));flash\_surface.set\_alpha(move\_flash\*20);flash\_surface.fill((255,255,255));screen.blit(flash\_surface,(0,0));move\_flash-=1

draw\_board\_and\_objects(screen,game\_state.current\_board,assets,board\_offset,target\_mask=game\_state.target\_mask,player\_direction=game\_state.player\_direction)

screen.blit(level\_text,text\_rect);draw\_header\_button(screen,nav\_buttons['menu'],'II',assets);draw\_header\_button(screen,nav\_buttons['prev'],'<',assets,enabled=current\_level\_idx>0);draw\_header\_button(screen,nav\_buttons['next'],'>',assets,enabled=current\_level\_idx<len(levels\_list)-1)

for name,rect in bottom\_buttons.items():draw\_header\_button(screen,rect,name,assets,enabled=not(name=="Solve" and not game\_state.solution))

elapsed\_time=time.time()-(game\_state.win\_time if game\_state.is\_won else game\_state.start\_time);moves\_count=len(game\_state.move\_stack)-1;score\_text=f"Moves: {moves\_count} | Time: {int(elapsed\_time)}";score\_surf=assets.font\_small.render(score\_text,True,assets.theme['TEXT']);screen.blit(score\_surf,score\_surf.get\_rect(right=W-20,bottom=H-20))

if game\_state.is\_won:

win\_text=assets.font\_large.render("Level Complete!",True,assets.theme['WIN']);screen.blit(win\_text,win\_text.get\_rect(centerx=W//2,bottom=H-80))

player\_stars=game.get\_stars\_for\_player(game.CURRENT\_PLAYER\_NAME)

for s in range(3):star\_center=(W//2-40+s\*40,H-45);color=assets.theme['STAR'] if s<player\_stars.get(game\_state.level\_key,0) else(80,90,100);draw\_star(screen,star\_center,20,color)

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)

player\_animator=None

async def level\_editor(screen,assets):

await fade\_transition(screen,assets,fade\_in=True);W,H=screen.get\_size();editor\_w,editor\_h=20,15;ts=assets.config.TILE\_SIZE

board=np.full((editor\_h,editor\_w),game.GameObject.EMPTY.value,dtype=int)

paint\_tools=[game.GameObject.WALL,game.GameObject.EMPTY];stamp\_tools=[game.GameObject.TARGET,game.GameObject.BOX,game.GameObject.PLAYER]

current\_paint\_tool=game.GameObject.WALL;held\_stamp=None

palette\_w=200;board\_offset\_x=(W-palette\_w-(editor\_w\*ts))//2;board\_offset\_y=(H-(editor\_h\*ts))//2;palette\_x=board\_offset\_x+(editor\_w\*ts)+20

all\_tools=paint\_tools+stamp\_tools

tool\_rects={tool:pygame.Rect(palette\_x,100+i\*70,60,60) for i,tool in enumerate(all\_tools)}

save\_btn=pygame.Rect(palette\_x,H-220,180,50);back\_btn=pygame.Rect(palette\_x,H-80,180,50)

mouse\_down=False;message="";message\_color='red';message\_timer=0;clock=pygame.time.Clock();

while True:

mouse\_pos=pygame.mouse.get\_pos()

if message\_timer>0:message\_timer-=1

for event in pygame.event.get():

if event.type==pygame.QUIT:return "QUIT"

if event.type==pygame.MOUSEBUTTONDOWN and event.button==1:

mouse\_down=True

if back\_btn.collidepoint(mouse\_pos):await fade\_transition(screen,assets,fade\_in=False);return "MAIN\_MENU"

if save\_btn.collidepoint(mouse\_pos):

p\_count=np.count\_nonzero(board==game.GameObject.PLAYER.value);b\_count=np.count\_nonzero(board==game.GameObject.BOX.value);t\_count=np.count\_nonzero(board==game.GameObject.TARGET.value)

if p\_count==1 and b\_count>0 and b\_count==t\_count:

if game.solve\_new\_level(board.tolist()):

game.save\_custom\_level(board.tolist());await fade\_transition(screen,assets,fade\_in=False);return "MODE\_SELECT"

else:message="Unsolvable!";message\_color=assets.theme['EDITOR\_MSG\_BAD'];message\_timer=120

else:message="Invalid Layout!";message\_color=assets.theme['EDITOR\_MSG\_BAD'];message\_timer=120

clicked\_on\_board=True

for tool,rect in tool\_rects.items():

if rect.collidepoint(mouse\_pos):

clicked\_on\_board=False

if tool in stamp\_tools:held\_stamp=tool;assets.sounds['button'].play()

else:current\_paint\_tool=tool;held\_stamp=None;assets.sounds['button'].play()

if clicked\_on\_board:

grid\_col=(mouse\_pos[0]-board\_offset\_x)//ts;grid\_row=(mouse\_pos[1]-board\_offset\_y)//ts

if 0<=grid\_row<editor\_h and 0<=grid\_col<editor\_w:

if held\_stamp is not None:

if held\_stamp==game.GameObject.PLAYER:board[board==game.GameObject.PLAYER.value]=game.GameObject.EMPTY.value

board[grid\_row,grid\_col]=held\_stamp.value;held\_stamp=None;assets.sounds['place\_box'].play()

else:board[grid\_row,grid\_col]=current\_paint\_tool.value;assets.sounds['move'].play()

if event.type==pygame.MOUSEBUTTONUP and event.button==1:mouse\_down=False

if event.type==pygame.MOUSEMOTION and mouse\_down and held\_stamp is None:

grid\_col=(mouse\_pos[0]-board\_offset\_x)//ts;grid\_row=(mouse\_pos[1]-board\_offset\_y)//ts

if 0<=grid\_row<editor\_h and 0<=grid\_col<editor\_w:

board[grid\_row,grid\_col]=current\_paint\_tool.value;assets.sounds['move'].play()

if event.type==pygame.KEYDOWN and event.key==pygame.K\_ESCAPE:await fade\_transition(screen,assets,fade\_in=False);return "MAIN\_MENU"

screen.fill(assets.theme['BG']);

if 'background' in assets.images:screen.blit(assets.images['background'],(0,0))

draw\_board\_and\_objects(screen,board,assets,(board\_offset\_x,board\_offset\_y),is\_editor=True)

p\_count=np.count\_nonzero(board==game.GameObject.PLAYER.value);b\_count=np.count\_nonzero(board==game.GameObject.BOX.value);t\_count=np.count\_nonzero(board==game.GameObject.TARGET.value)

valid\_level=p\_count==1 and b\_count>0 and b\_count==t\_count

if p\_count!=1:help\_text=f"Needs {1-p\_count} Player"

elif b\_count==0:help\_text="Needs Boxes"

elif b\_count!=t\_count:help\_text=f"{abs(b\_count-t\_count)} more {'Targets' if b\_count>t\_count else 'Boxes'}"

else:help\_text="Ready to Test!"

help\_surf=assets.font\_small.render(help\_text,True,assets.theme['WIN'] if valid\_level else assets.theme['EDITOR\_TEXT'])

screen.blit(help\_surf,help\_surf.get\_rect(centerx=palette\_x+90,y=H-260))

if message\_timer>0:msg\_surf=assets.font\_small.render(message,True,message\_color);screen.blit(msg\_surf,msg\_surf.get\_rect(centerx=palette\_x+90,y=H-280))

for tool,rect in tool\_rects.items():

image\_key={game.GameObject.WALL:'wall',game.GameObject.EMPTY:'floor',game.GameObject.TARGET:'target',game.GameObject.BOX:'box',game.GameObject.PLAYER:'player\_face'}.get(tool)

is\_toggled=(tool in paint\_tools and tool==current\_paint\_tool and held\_stamp is None) or (tool==held\_stamp)

draw\_header\_button(screen,rect,"",assets,toggled=is\_toggled)

if image\_key in assets.images:screen.blit(pygame.transform.smoothscale(assets.images[image\_key],(50,50)),(rect.x+5,rect.y+5))

if held\_stamp:

stamp\_image\_key={game.GameObject.TARGET:'target',game.GameObject.BOX:'box',game.GameObject.PLAYER:'player\_front'}.get(held\_stamp)

if stamp\_image\_key in assets.images:sprite=assets.images[stamp\_image\_key];sprite.set\_alpha(150);screen.blit(sprite,(mouse\_pos[0]-ts//2,mouse\_pos[1]-ts//2));sprite.set\_alpha(255)

draw\_header\_button(screen,save\_btn,"SAVE & TEST",assets,enabled=valid\_level);draw\_header\_button(screen,back\_btn,"BACK",assets)

pygame.display.flip();await asyncio.sleep(0);clock.tick(assets.config.FPS)