**GAME DESIGN DOCUMENT**

**[Jumping Physics Puzzle]**

Course Title: 2D & 3D Game Development using Pygame

Submitted by:

Kallem Laxminarayana Team Titte: Tarun Chilukuri

231FA04B93

Pachipulusu Abhilash

231FA04344

Chilukuri Tarun kumar reddy

231FA04C90

Polamreddy Kowshik reddy

231FA04F34

Vignan university

Date:26-5-25

**Simple Platformer: Core Mechanics**

* Overview of a basic cube-jumping platformer game.
* Focus on core mechanics: gravity, jumping, collision.

**Gravity Implementation**

* Constant downward acceleration (e.g., -10 units/frame).
* Velocity updates each frame: velocity += gravity.
* Terminal velocity limit (e.g., -20 units/frame).
* Example: velocity = max(velocity, -20);

**Jumping Logic**

* Initial upward velocity (e.g., +15 units/frame).
* Condition for jump: cube is on a platform.
* Jump input resets upward velocity.
* Double jump: implement a jump counter.
* Jump sustain with upward thrust that decays.

**Platform Collision**

* Axis-aligned bounding box (AABB) collision.
* Check for overlap between cube and platform rects.
* Resolve collision by moving cube above the platform.
* Prevents clipping; enables "grounded" state.
* Collision direction: calculate dot product of normals.

**Platform Generation**

* Random platform placement within bounds.
* Minimum and maximum platform distances.
* Gap width control: adjusts difficulty.
* Platform variation: static, moving (horizontally, vertically).
* Platform types: normal, bouncy, ice.

**Target Implementation**

* Target rectangle at the end of the level.
* Visual cue: different color, animation.
* Target position fixed or moving.
* Collectible before target: Key increases score by 100.

**Win Condition**

* Cube collides with target rectangle.
* Display "You Win!" message.
* Stop player movement.
* Score: distance travelled + time remaining \* 10.

**Level Design Considerations**

* Increasing difficulty: platform speed, gaps.
* Visual appeal: background, color schemes.
* Introduce moving or disappearing platforms.
* Add obstacles: projectiles, enemies.
* Tutorial level: guide the player, introduce mechanics.

Project code:

import pygame

import sys

pygame.init()

WIDTH, HEIGHT = 800, 600

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

pygame.display.set\_caption("Simple Platformer")

WHITE = (255, 255, 255)

BLUE = (0, 0, 255)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLACK = (0, 0, 0)

player\_size = 30

player\_x = 50

player\_y = HEIGHT - player\_size - 50

player\_vel\_y = 0

player\_jump = False

gravity = 1

jump\_strength = -15

player\_speed = 5

platforms = [

pygame.Rect(0, HEIGHT - 20, WIDTH, 20), # Ground

pygame.Rect(200, HEIGHT - 100, 100, 20),

pygame.Rect(350, HEIGHT - 200, 100, 20),

pygame.Rect(500, HEIGHT - 300, 100, 20),

pygame.Rect(650, HEIGHT - 200, 100, 20)

]

target = pygame.Rect(700, HEIGHT - 250, 30, 30)

game\_won = False

clock = pygame.time.Clock()

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_SPACE and not player\_jump:

player\_vel\_y = jump\_strength

player\_jump = True

if not game\_won:

keys = pygame.key.get\_pressed()

if keys[pygame.K\_LEFT]:

player\_x -= player\_speed

if keys[pygame.K\_RIGHT]:

player\_x += player\_speed

player\_vel\_y += gravity

player\_y += player\_vel\_y

player\_rect = pygame.Rect(player\_x, player\_y, player\_size, player\_size)

player\_jump = True

for platform in platforms:

if player\_rect.colliderect(platform) and player\_vel\_y > 0:

if player\_y + player\_size <= platform.y + 10:

player\_y = platform.y - player\_size

player\_jump = False

player\_vel\_y = 0

if player\_y > HEIGHT:

player\_x = 50

player\_y = HEIGHT - player\_size - 50

player\_vel\_y = 0

if player\_rect.colliderect(target):

game\_won = True

screen.fill(BLACK)

for platform in platforms:

pygame.draw.rect(screen, BLUE, platform)

pygame.draw.rect(screen, GREEN if game\_won else RED, target)

pygame.draw.rect(screen, WHITE, (player\_x, player\_y, player\_size, player\_size))

if game\_won:

font = pygame.font.SysFont(None, 72)

text = font.render("You Win!", True, WHITE)

screen.blit(text, (WIDTH//2 - text.get\_width()//2, HEIGHT//2 - text.get\_height()//2))

pygame.display.flip()

clock.tick(60)

pygame.quit()

sys.exit()

Output:

