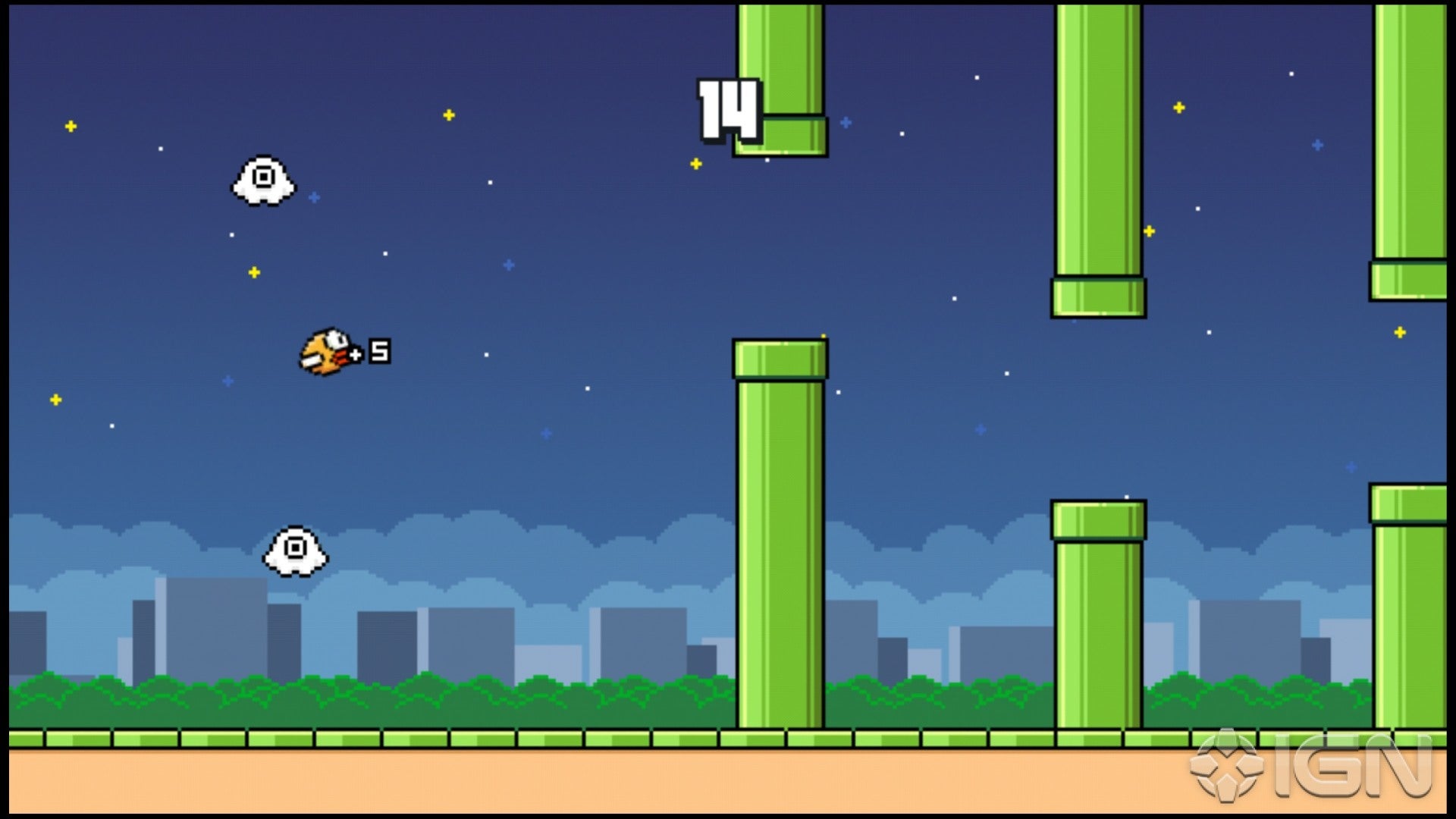
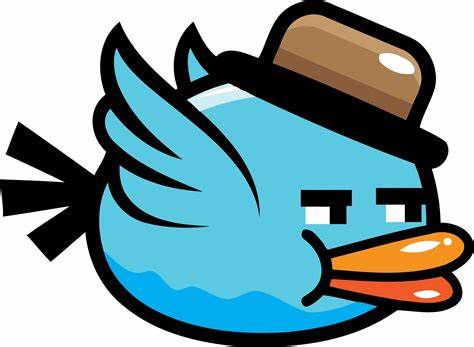
1. **Page de garde :**
   * Titre du jeu : Birdy Fly
   * Logo du jeu (le cas échéant) :
   * Noms des membres de l'équipe : Pauline Calteau, Kénya Hoareau
   * Date de création de la version du GDD : 29/02/2024
2. **Sommaire :**
   * Liste des sections et sous-sections avec les numéros de page correspondants.
3. **Introduction :**
   * Présentation générale du jeu : C’est un jeu inspiré de Flappy Bird.
   * Objectifs du jeu : passer entre deux obstacles en volant.
   * Public cible : Tout public
4. **Concept du jeu :**
   * Résumé du scénario (le cas échéant)
   * Thème du jeu : plateformer
   * Style artistique et visuel : jeu 2D
5. **Gameplay :**
   * Mécaniques de jeu :
   * Systèmes de contrôle : barre espace et souris pour naviguer
   * Éléments de jeu : petit oiseau ; des arbres et rochers.
   * Progression du joueur
   * Niveaux ou mondes du jeu : Il y a 3 niveau le niveau 3 est le plus compliqué, easy hard, hardcore.
6. **Monétisation (le cas échéant) :**
   * Modèle économique : jeu gratuit
   * Achats intégrés : non
   * Publicités (le cas échéant) : oui
7. **Interface utilisateur (UI) :**
   * Conception des menus :
   * HUD (Head-Up Display) : rond parametres, le play, quitter
   * Éléments de l'interface utilisateur :
8. **Audio :**
   * Musique : oui
   * Effets sonores : bruits de battement d’ailes
   * Doublages (le cas échéant)
9. **Graphismes :**
   * Style artistique
   * Conception des personnages
   * Environnements et niveaux
10. **Technologie :**
    * Plateforme cible : PC et mobile
    * Moteur de jeu utilisé : Pygame
    * Spécifications techniques
11. **Tests et debug :**
    * Plan de test
    * Procédures de débogage
    * Rapports de bugs
12. **Marketing et promotion :**
    * Stratégie de marketing
    * Campagnes publicitaires
    * Réseaux sociaux
13. **Échéancier de développement :**
    * Liste des tâches
    * Dates limites
    * Milestones
14. **Budget (le cas échéant) :**
    * Coûts de développement
    * Marketing
    * Autres dépenses
15. **Annexes :**
    * Diagrammes, croquis, captures d'écran, etc.



Lien github code m a j : [Kenyawaro/Birdy-Fly (github.com)](https://github.com/Kenyawaro/Birdy-Fly)

import pygame

import random

# Initialisation de Pygame

pygame.init()

# Paramètres de la fenêtre

WINDOW\_WIDTH = 800

WINDOW\_HEIGHT = 600

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

pygame.display.set\_caption("Geometry Dash")

# Couleurs

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

# Variables du joueur

player\_width = 50

player\_height = 50

player\_x = 50

player\_y = WINDOW\_HEIGHT // 2 - player\_height // 2

player\_velocity = 10

# Variables des obstacles

obstacle\_width = 50

obstacle\_height = random.randint(50, 300)

obstacle\_x = WINDOW\_WIDTH

obstacle\_y = WINDOW\_HEIGHT - obstacle\_height

obstacle\_velocity = 10

obstacle\_gap = 200

# Score

score = 0

font = pygame.font.Font(None, 36)

# Fonction pour afficher le score

def display\_score():

score\_text = font.render("Score: " + str(score), True, BLACK)

WINDOW.blit(score\_text, (10, 10))

# Boucle principale du jeu

clock = pygame.time.Clock()

running = True

while running:

WINDOW.fill(WHITE)

# Gestion des événements

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Mouvement du joueur

keys = pygame.key.get\_pressed()

if keys[pygame.K\_SPACE]:

player\_y -= player\_velocity

else:

player\_y += player\_velocity

# Mouvement de l'obstacle

obstacle\_x -= obstacle\_velocity

# Gestion des collisions

if player\_x + player\_width > obstacle\_x and player\_x < obstacle\_x + obstacle\_width:

if player\_y < obstacle\_height or player\_y + player\_height > obstacle\_height + obstacle\_gap:

running = False

else:

score += 1

# Affichage du joueur

pygame.draw.rect(WINDOW, BLACK, (player\_x, player\_y, player\_width, player\_height))

# Affichage de l'obstacle

pygame.draw.rect(WINDOW, BLACK, (obstacle\_x, 0, obstacle\_width, obstacle\_height))

pygame.draw.rect(WINDOW, BLACK, (obstacle\_x, obstacle\_height + obstacle\_gap, obstacle\_width, WINDOW\_HEIGHT - obstacle\_height - obstacle\_gap))

# Réinitialisation de l'obstacle lorsqu'il sort de l'écran

if obstacle\_x + obstacle\_width < 0:

obstacle\_x = WINDOW\_WIDTH

obstacle\_height = random.randint(50, 300)

score += 1

# Affichage du score

display\_score()

pygame.display.update()

clock.tick(30)

# Fermeture de Pygame

pygame.quit()

DERNIER BON CODE

import pygame

import random

# Initialisation de Pygame

pygame.init()

# Paramètres de la fenêtre

WINDOW\_WIDTH = 800

WINDOW\_HEIGHT = 600

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

pygame.display.set\_caption("Geometry Dash")

# Chargement de l'image de l'arrière-plan

background\_img = pygame.image.load("IMG\_3123.png").convert()  # Assurez-vous que le chemin d'accès à l'image est correct

# Chargement de l'image du joueur

player\_img = pygame.image.load("IMG\_3125-\_1\_-removebg-preview.png").convert\_alpha()

# Couleurs

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

# Variables du joueur

player\_width = 50

player\_height = 50

player\_x = 50

player\_y = WINDOW\_HEIGHT // 2 - player\_height // 2

player\_velocity = 10

# Variables des obstacles

obstacle\_width = 50

obstacle\_gap = 200

obstacle\_velocity = 5

# Liste pour stocker les obstacles

obstacles = []

# Score

score = 0

font = pygame.font.Font(None, 36)

# Fonction pour afficher le score

def display\_score():

    score\_text = font.render("Score: " + str(score), True, BLACK)

    WINDOW.blit(score\_text, (10, 10))

# Fonction pour créer un nouvel obstacle

def create\_obstacle():

    obstacle\_height = random.randint(50, WINDOW\_HEIGHT - obstacle\_gap - 50)

    top\_obstacle\_rect = pygame.Rect(WINDOW\_WIDTH, 0, obstacle\_width, obstacle\_height)

    bottom\_obstacle\_rect = pygame.Rect(WINDOW\_WIDTH, obstacle\_height + obstacle\_gap, obstacle\_width, WINDOW\_HEIGHT - obstacle\_height - obstacle\_gap)

    obstacles.append((top\_obstacle\_rect, bottom\_obstacle\_rect))

# Création du premier obstacle

create\_obstacle()

# Boucle principale du jeu

clock = pygame.time.Clock()

running = True

while running:

    WINDOW.blit(background\_img, (0, 0))  # Dessiner l'arrière-plan

    # Gestion des événements

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            running = False

    # Mouvement du joueur

    keys = pygame.key.get\_pressed()

    if keys[pygame.K\_SPACE]:

        player\_y -= player\_velocity

    else:

        player\_y += player\_velocity

    # Vérification si le joueur touche le sol

    if player\_y + player\_height >= WINDOW\_HEIGHT:

        running = False  # Le joueur meurt s'il touche le sol

    # Vérification si le joueur touche un obstacle

    for top\_obstacle, bottom\_obstacle in obstacles:

        if (top\_obstacle.colliderect(player\_x, player\_y, player\_width, player\_height) or

                bottom\_obstacle.colliderect(player\_x, player\_y, player\_width, player\_height)):

            running = False

    # Mouvement des obstacles

    for top\_obstacle, bottom\_obstacle in obstacles:

        top\_obstacle.x -= obstacle\_velocity

        bottom\_obstacle.x -= obstacle\_velocity

    # Suppression des obstacles qui sont sortis de l'écran

    obstacles = [(top\_obstacle, bottom\_obstacle) for top\_obstacle, bottom\_obstacle in obstacles if top\_obstacle.x + obstacle\_width > 0]

    # Création d'un nouvel obstacle si nécessaire

    if obstacles[-1][0].x < WINDOW\_WIDTH - 300:

        create\_obstacle()

    # Affichage du joueur

    WINDOW.blit(player\_img, (player\_x, player\_y))

    # Affichage des obstacles

    for top\_obstacle, bottom\_obstacle in obstacles:

        pygame.draw.rect(WINDOW, BLACK, top\_obstacle)

        pygame.draw.rect(WINDOW, BLACK, bottom\_obstacle)

    # Affichage du score

    display\_score()

    pygame.display.update()

    clock.tick(30)

# Fermeture de Pygame

pygame.quit()

BON CODE

import pygame

import random

# Initialisation de Pygame

pygame.init()

# Paramètres de la fenêtre

WINDOW\_WIDTH = 800

WINDOW\_HEIGHT = 600

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

pygame.display.set\_caption("Geometry Dash")

# Chargement de l'image de l'arrière-plan

background\_img = pygame.image.load("d:\Perso\codage\IMG\_3123.JPG ").convert()  # Assurez-vous que le chemin d'accès à l'image est correct

# Couleurs

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

# Variables du joueur

player\_width = 50

player\_height = 50

player\_x = 50

player\_y = WINDOW\_HEIGHT // 2 - player\_height // 2

player\_velocity = 10

# Variables des obstacles

obstacle\_width = 50

obstacle\_height = random.randint(50, 300)

obstacle\_x = WINDOW\_WIDTH

obstacle\_y = WINDOW\_HEIGHT - obstacle\_height

obstacle\_velocity = 10

obstacle\_gap = 200

# Score

score = 0

font = pygame.font.Font(None, 36)

# Fonction pour afficher le score

def display\_score():

    score\_text = font.render("Score: " + str(score), True, BLACK)

    WINDOW.blit(score\_text, (10, 10))

# Création des rectangles de collision pour le joueur et les obstacles

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

obstacle\_rect1 = pygame.Rect(obstacle\_x, 0, obstacle\_width, obstacle\_height)

obstacle\_rect2 = pygame.Rect(obstacle\_x, obstacle\_height + obstacle\_gap, obstacle\_width, WINDOW\_HEIGHT - obstacle\_height - obstacle\_gap)

# Boucle principale du jeu

clock = pygame.time.Clock()

running = True

obstacle\_passed = False  # Variable pour suivre si le joueur a passé un obstacle

while running:

    WINDOW.blit(background\_img, (0, 0))  # Dessiner l'arrière-plan

    # Gestion des événements

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            running = False

    # Mouvement du joueur

    keys = pygame.key.get\_pressed()

    if keys[pygame.K\_SPACE]:

        player\_y -= player\_velocity

    else:

        player\_y += player\_velocity

    # Vérification si le joueur touche le sol

    if player\_y + player\_height >= WINDOW\_HEIGHT:

        running = False  # Le joueur meurt s'il touche le sol

    # Mouvement de l'obstacle

    obstacle\_x -= obstacle\_velocity

    # Mise à jour des rectangles de collision

    player\_rect.y = player\_y

    obstacle\_rect1.x = obstacle\_x

    obstacle\_rect2.x = obstacle\_x

    # Gestion des collisions

    if player\_rect.colliderect(obstacle\_rect1) or player\_rect.colliderect(obstacle\_rect2):

        running = False

    # Gestion du score

    if obstacle\_x + obstacle\_width < player\_x and not obstacle\_passed:

        score += 1

        obstacle\_passed = True

    # Affichage du joueur

    pygame.draw.rect(WINDOW, BLACK, player\_rect)

    # Affichage de l'obstacle

    pygame.draw.rect(WINDOW, BLACK, obstacle\_rect1)

    pygame.draw.rect(WINDOW, BLACK, obstacle\_rect2)

    # Réinitialisation de l'obstacle lorsqu'il sort de l'écran

    if obstacle\_x + obstacle\_width < 0:

        obstacle\_x = WINDOW\_WIDTH

        obstacle\_height = random.randint(50, 300)

        obstacle\_passed = False

    # Affichage du score

    display\_score()

    pygame.display.update()

    clock.tick(30)

# Fermeture de Pygame

pygame.quit()