

Helwan University

Faculty of Computers & Artificial Intelligence

Computer Science Department

[Facial Expression Game]



A graduation project Dissertation By:

Abd El-Rahman Hamdy Abd El-Aty [202000510]

Abdullah Shaaban Abd El-Razek [202000548]

Shهاب El-Din Mokhtar El-Said [202000441]

Abd El-Rahman Yasser Samir [202000542]

Youssef Mohamed Sayed [202001105]

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University.

Supervised by:

Dr. Mohammed El-Said

June 2024



كلية الحاسبات والذكاء الاصطناعي
Faculty of Computers & Artificial Intelligence



جامعة حلوان

كلية الحاسبات والذكاء الاصطناعي

قسم علوم الحاسب

[لعبة تعابير الوجه]



رسالة مشروع تخرج مقدمة من:

عبد الرحمن حمدي عبد العاطي [202000510]

عبد الله شعبان عبد الرازق [202000548]

شهاب الدين مختار السعيد [202000441]

عبد الرحمن ياسر سمير [202000542]

يوسف محمد سيد محمد [202001105]

رسالة مقدمة ضمن متطلبات الحصول على درجة البكالوريوس في الحاسبات والذكاء

الاصطناعي بقسم علوم الحاسب، كلية الحاسبات

والذكاء الاصطناعي، جامعة حلوان

تحت إشراف :

د محمد السعيد

يونيو / حزيران 2024

Facial Expression Game



Abstract

Facial expression games represent a novel intersection of interactive entertainment and advanced computer vision technology. This graduation project explores the development of such games using the Unity game engine and facial expression recognition techniques, specifically focusing on detecting and utilizing happy expressions. Our objective is to create immersive and intuitive gameplay experiences that respond dynamically to the player's emotional state, aiming to improve the player's mood.

Our project utilizes facial expression recognition to detect and interpret a player's happy expressions in real-time. This data is then integrated into the Unity engine to influence game mechanics, offering a unique and engaging way for players to interact with the game environment. The methodology involves training a facial recognition model to accurately identify happiness and subsequently mapping these expressions to corresponding in-game actions and events. By doing so, we aim to create games that adapt to the player's emotional state, providing positive reinforcement and mood-enhancing experiences.

Through this project, we demonstrate the potential of happy expression-based interaction as a viable and innovative input method for video games. The implementation details, challenges encountered, and solutions devised during the development process are thoroughly documented. Also, we present an analysis of user feedback and performance metrics to evaluate the effectiveness and enjoyment of the happy expression control system and its impact on the player's mood.

This project not only contributes to the field of game design but also paves the way for further exploration into emotion-sensitive applications in various domains. The implications of such technology extend beyond gaming, offering possibilities

For enhanced user experiences in virtual reality, interactive storytelling, and therapeutic applications.

* *

Keywords

- Facial Expression Recognition
- Unity Game Engine
- Computer Vision
- Emotional Interaction
- Happy Expression Detection
- Mood Enhancement
- Real-time Emotion Detection
- Interactive Gameplay
- User Experience
- Game Development
- Player Engagement
- Positive Reinforcement
- Emotion-Sensitive Applications

.....

Acknowledgment

To be able to finish this project successfully, we were not alone, but we would like to thank everyone who contributed to its successful completion.

First and foremost, we would like to thank our supervisor, **Dr. Mohammed El-Said**, for his invaluable guidance, continuous support, and encouragement throughout the development of this project. His expertise and insights have been instrumental in shaping the direction and outcome of this work.

Our team members, whose dedication and collaborative spirit made this project possible. Their hard work, creativity, and commitment to excellence were essential in overcoming the challenges we faced and achieving our goals.

A special thanks to **Dr. Heba Labna**, from the Department of Psychology, Faculty of Art, Helwan University, for her assistance in collecting and interpreting the psychological evidence relevant to this project. Her expertise and support were crucial in integrating the psychological aspects into our work.

Together, we have created a project that we can all be proud of, and we truly grateful for the opportunity to work with such talented and motivated individuals.

Thank you all for your contributions, support, and encouragement.

* *

Glossary

Artificial Intelligence (AI): A field of computer science that emphasizes the creation of intelligent machines that work and react like humans.

CNN (Convolutional Neural Network): A type of deep neural network commonly used for analyzing visual imagery, which is well-suited for image classification and recognition tasks.

Computer Vision: A field of artificial intelligence that trains computers to interpret and understand the visual world. Using digital images from cameras and videos, and deep learning models, machines can accurately identify and classify objects.

Dataset: A collection of data used to train and evaluate machine learning models. In this project, FER 2013 is used, which contains images of facial expressions.

Facial Expression Recognition: A technology capable of detecting and interpreting human facial expressions to understand the emotional state of the person.

FER 2013: A publicly available dataset consisting of labeled facial expressions, used for training and testing facial expression recognition models.

AffectNet: AffectNet is a large-scale dataset that contains images annotated with facial expressions. For this system, a subset of AffectNet with either 2 or 8 facial expressions was used, depending on the experiment.

Game Engine: Software used for the development of video games. Unity is a popular game engine used in this project.

Machine Learning: A subset of artificial intelligence that involves training algorithms to make predictions or decisions based on data.

Model Accuracy: A metric that measures the percentage of correct predictions made by a model. Training accuracy refers to performance on the training dataset, while validation accuracy refers to performance on unseen data.

Neural Network: A series of algorithms that attempt to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

Positive Reinforcement: A concept in psychology where behaviors are encouraged through rewards or positive outcomes, used in this project to enhance player mood.

Real-time Processing: The capability of a system to process data and provide outputs almost instantaneously.

Unity: A cross-platform game engine used for developing video games and simulations. It supports 2D and 3D graphics, drag-and-drop functionality, and scripting through C#.

User Experience (UX): The overall experience of a person using a product, especially in terms of how easy or pleasing it is to use.

Validation Dataset: A subset of data used to assess the performance of a model during training, to ensure it generalizes well to new, unseen data.

* *

Table of Contents

- Abstract..... 4
- Keywords..... 5
- Acknowledgment 6
- Glossary..... 7
- Chapter 1..... 10
 - An Introduction 10
 - 1.1 Overview 11
 - 1.2 Problem Statement 14
 - 1.3 Scope and Objectives 16
 - 1.4 Work Methodology 18
- Chapter 2..... 21
 - Literature Review 21
 - 2.1 Background 22
 - 2.2 Literature Survey on the Multifaceted Nature of Smiling and Its Impact on Communication, Well-being, and Player Mood in Gaming..... 24
- Chapter 3..... 26
 - Implementation, Experimental Setup, Results 26
 - 3.1 Implementation details 27
- Chapter 4..... 56
 - Discussion, Conclusions, and Future Work 56
 - 4.1 Discussion..... 57
 - 4.2 Summary & Conclusion 57
 - 4.3 Future Work 58
- Appendix 60
 - A. Game Screenshots 61
 - D. Tools and Technology..... 70

Chapter 1

An Introduction

1.1 Overview

1.1.1 *Introduction*

Facial expression games represent a cutting-edge integration of interactive entertainment and advanced computer vision technology. This project focuses on developing games that utilize facial expression recognition, specifically detecting happy and unhappy expressions, to create engaging and mood-enhancing gameplay experiences. The integration of this technology into game design offers a novel approach to user interaction, making the gaming experience more immersive and emotionally responsive.

1.1.2 *Objectives*

The primary objective of this project is to design and implement a game that responds to the player's happy facial expressions using the Unity game engine and a MobileNet architecture enhanced with additional fully connected layers. By doing so, we aim to:

- Enhance the player's mood through positive reinforcement.
- Provide an innovative and intuitive interaction method.
- Explore the potential of emotion-sensitive applications in gaming.

1.1.3 Methodology

The project employs the AffectNet dataset, which contains images of various facial expressions. For simplicity and focus, only happy expressions were considered as happy, and all other expressions were grouped as unhappy. The methodology involves several key steps:

1. **Data Collection and Preprocessing:** Gathering and preparing the AffectNet dataset, including under-sampling to balance the target classes.
2. **Model Training:** Developing and training a MobileNet architecture with additional fully connected layers to recognize happy and unhappy expressions.
3. **Integration with Unity:** Implementing the trained model within the Unity game engine to create interactive gameplay elements that respond to the player's expressions.
4. **User Testing and Feedback:** Conducting user tests to evaluate the game's effectiveness and enjoyment and its impact on the player's mood.

1.1.4 Implementation

The implementation phase involves the following components:

- **Facial Expression Recognition:** Using computer vision techniques to detect and interpret happy and unhappy expressions in real-time.
- **Unity Game Development:** Creating 2 games environment where in-game actions and events are dynamically influenced by the player's facial expressions.
- **Positive Reinforcement Mechanisms:** Incorporating elements that adapt to the player's happy expressions, enhancing the overall gaming experience and mood.

1.1.5 Results and Analysis

The project results demonstrate the feasibility and effectiveness of using facial expression recognition to enhance user interaction in games. The custom MobileNet model, enhanced with additional fully connected layers, achieved a training accuracy of 92% and a validation accuracy of 91%. User feedback indicated a positive reception to the

innovative interaction method, with many players reporting an improvement in mood and a more engaging gaming experience.

1.1.6 Conclusion

This project successfully showcases the potential of facial expression-based interaction as a viable input method for video games. The integration of computer vision and game design not only enhances user experience but also opens new possibilities for emotion-sensitive applications in various domains, including virtual reality, interactive storytelling, and therapeutic applications. Future work will focus on refining the model, exploring additional expressions, and expanding the application scope.

* *

1.2 Problem Statement

In the realm of video gaming, user interaction has traditionally been limited to physical inputs such as keyboard, mouse, and game controllers. While these methods provide a rich array of controls, they lack the capacity to fully engage the emotional state of the player. Current gaming experiences do not dynamically adapt to the player's emotional state, potentially leading to a disconnect between the player and the game and missing an opportunity to enhance the player's overall experience and mood.

Moreover, the gaming industry has seen a growing interest in creating more immersive and interactive environments. However, integrating emotional feedback into game design remains a relatively unexplored area. Games that can recognize and respond to the emotional expressions of players could significantly enhance user engagement and satisfaction.

Our project addresses this gap by focusing on the development of a game that uses facial expression recognition to identify happy and unhappy expressions. The primary challenge is to accurately detect these expressions in real-time and integrate this feedback into the game environment to create a more immersive and emotionally responsive experience. This approach aims to improve player engagement and enhance mood, providing a novel and intuitive method of interaction.

The specific problems addressed in this project include:

- **Emotion Detection Accuracy:** Developing a robust model capable of accurately detecting and classifying happy and unhappy expressions using the AffectNet dataset.
- **Real-Time Processing:** Ensuring that the facial expression recognition system operates in real-time to provide immediate feedback and interaction within the game.
- **Game Integration:** Seamlessly integrating the emotion detection model with the Unity game engine to create dynamic and responsive gameplay elements.
- **Mood Enhancement:** Designing game mechanics that positively reinforce happy expressions, thereby improving the player's mood and overall gaming experience.

By addressing these challenges, the project seeks to demonstrate the potential of emotion-sensitive interaction in gaming, paving the way for future innovations in this field.

* *

1.3 Scope and Objectives

1.3.1 Scope

This project focuses on the development of a facial expression-based game using Unity and computer vision techniques. The scope of the project includes:

1. Facial Expression Recognition:

- Utilizing the AffectNet dataset to train a model capable of recognizing happy and unhappy expressions.
- Implementing a custom MobileNet architecture with additional fully connected layers to achieve high accuracy in emotion detection.

2. Model Conversion and Integration:

- Using the Barracuda package to enable deep learning inference within Unity.
- Converting the trained model from .h5 format to ONNX (Open Neural Network Exchange) format for compatibility with Unity.

3. Game Development:

- Designing and developing a game in Unity that integrates facial expression recognition to influence gameplay.
- Creating game mechanics that adapt to the player's detected expressions, particularly focusing on promoting and responding to happy expressions.

4. Real-Time Interaction:

- Ensuring the emotion detection system operates in real-time to provide immediate feedback and interaction within the game environment.

5. User Experience and Mood Enhancement:

- Evaluating the impact of facial expression-based interaction on user engagement and mood.
- Collecting and analyzing user feedback to assess the effectiveness and enjoyment of the game.

1.3.2 Objectives

The primary objectives of this project are:

- 1. Develop a Facial Expression Recognition Model:**
 - Train a robust MobileNet-based model using the AffectNet dataset to accurately detect happy and unhappy facial expressions.
 - Achieve a training accuracy of at least 92% and a validation accuracy of at least 91%.
- 2. Model Conversion and Unity Integration:**
 - Convert the trained model from .h5 format to ONNX format for compatibility with Unity.
 - Use the Barracuda package to implement the ONNX model in Unity, ensuring smooth and efficient deep learning inference.
- 3. Integrate Emotion Detection with Unity:**
 - Seamlessly incorporate the trained emotion detection model into the Unity game engine.
 - Develop game mechanics that dynamically respond to the player’s facial expressions.
- 4. Enhance Player Engagement and Mood:**
 - Design game elements that provide positive reinforcement for happy expressions to enhance the player’s mood.
 - Ensure the game provides an intuitive and enjoyable experience, encouraging continued play and engagement.
- 5. Document and Evaluate the Project:**
 - Thoroughly document the development process, including the challenges encountered and solutions implemented.
 - Evaluate the project's success based on user feedback, model performance, and the effectiveness of mood enhancement strategies.

By achieving these objectives, the project aims to demonstrate the potential of emotion-sensitive interaction in gaming, providing insights into its applications and benefits.

* *

1.4 Work Methodology

The development of the facial expression-based game involved several key phases: data collection and preprocessing, model development and training, model conversion and integration, game development, and user testing. Each phase was designed to ensure the accuracy, efficiency, and user engagement of the final product.

1.4.1 Data Collection and Preprocessing

Dataset Selection:

- **AffectNet Dataset:** We selected the AffectNet dataset for its comprehensive collection of facial expressions. For this project's purposes, we focused on images labeled with happy and unhappy expressions.

Data Generator:

To improve model generalization and prevent overfitting, we used a data generator to dynamically augment the training data during the training process. This approach increases the diversity of the training dataset by applying random transformations to the images on the fly. The data generator applied various techniques

Undersampling:

- We undersampled the dataset to balance the number of happy and unhappy expressions, ensuring that the model was not biased towards any particular class.

1.4.2 Model Development and Training

Model Architecture:

- **MobileNet Architecture:** We chose MobileNet for its efficiency and ability to perform well on mobile and embedded devices. The architecture was augmented with additional fully connected layers to enhance its capability to classify facial expressions accurately.

Training Process:

- **Training and Validation Split:** The dataset was divided into training and validation sets to evaluate the model's performance.
- **Training Parameters:** We used parameters such as learning rate, batch size, and number of epochs to train the model.
- **Optimization:** The Adam optimizer was used for efficient gradient-based optimization.

Performance Metrics:

- **Accuracy:** The primary metric for evaluating the model was accuracy, calculated for both training and validation sets.
-

1.4.3 Model Conversion and Integration

Model Conversion:

- **ONNX Format:** The trained model was converted from .h5 format to ONNX (Open Neural Network Exchange) format to facilitate integration with Unity.

Barracuda Integration:

- **Barracuda Package:** We used the Barracuda package in Unity for running deep learning inference. The ONNX model was integrated into Unity using this package, ensuring compatibility and efficient execution.

1.4.4 Game Development

Unity Game Engine:

- **Game Design:** The game was designed in Unity, focusing on creating a dynamic environment that responds to the player's facial expressions.
- **User Interface:** A user-friendly interface was developed to provide clear feedback to the player based on their detected expressions.

Gameplay Mechanics:

- **Emotion-Responsive Elements:** Game mechanics were designed to adapt to the player's happy expressions, providing positive reinforcement and enhancing the gaming experience.
- **Real-Time Interaction:** Ensuring real-time detection and response to the player's facial expressions was a critical aspect of the gameplay.

Chapter 2

Literature Review

2.1 Background

In recent years, the landscape of gaming has evolved significantly, moving beyond mere entertainment to encompass immersive experiences that engage players on emotional and psychological levels. Central to this evolution is the concept of emotion-sensitive interaction, which seeks to bridge the gap between players and game worlds by integrating the player's emotional state into gameplay mechanics. At the heart of emotion-sensitive interaction lies facial expression recognition (FER), a technology that enables computers to detect and interpret human emotions through facial expressions.

The roots of facial expression recognition can be traced back to early psychological research on emotion, particularly the work of pioneers such as Paul Ekman, who identified a universal set of facial expressions corresponding to basic emotions. Building upon this foundation, researchers in computer vision and artificial intelligence have developed techniques to automate the process of facial expression analysis, opening up new possibilities for applications in fields ranging from human-computer interaction to affective computing.

Facial expression is an important part of human communication. As a result, an inability to produce facial expressions leads to communication difficulties (Sawyer et al., August 2008). A smile, for instance, can express a whole host of emotions, ranging from the warmth of contentment, joy, delight, and affection to arrogance and bafflement (Sawyer et al., August 2008). Smiling is also identified as a highly effective means of nonverbal communication, and an inability to smile has a profoundly negative impact on quality of life (Dusseldorp et al., 2019).

Concurrently, the gaming industry has witnessed a growing interest in creating more immersive and emotionally resonant experiences. Game developers have begun exploring ways to leverage facial expression recognition and other emotion detection technologies to enhance gameplay, tailoring in-game experiences to match the player's emotional state. This trend has led to the emergence of emotion-sensitive games, which dynamically adapt to the player's emotions, offering personalized and engaging experiences.

Self-report measures showed that participants in the ‘enhanced smile’ condition felt more positive affect after the conversation and experienced stronger social presence compared to the ‘normal smile’ condition (Oh et al., September 2016). This suggests that enhancing facial expressions, even in virtual environments, can have significant effects on user experience and emotional engagement.

Against this backdrop, a body of literature has emerged, spanning research papers, academic journals, and industry publications, exploring various aspects of facial expression recognition, emotion-sensitive gaming, and the integration of computer vision techniques with game design. This literature review aims to synthesize and critically evaluate existing research and developments in these areas, providing insights into the current state of the field and informing the methodology and objectives of this project. Through a thorough examination of the literature, we seek to identify key trends, challenges, and opportunities, laying the groundwork for the development of a facial expression-based game that aims to enhance user experience and player engagement.

* *

2.2 Literature Survey on the Multifaceted Nature of Smiling and Its Impact on Communication, Well-being, and Player Mood in Gaming

Introduction

Smiling is a fundamental aspect of human communication, conveying a wide range of emotions and playing a crucial role in social interactions. This literature survey explores the multifaceted nature of smiling and its implications for communication, well-being, and player mood in gaming.

Smiling as a Complex Expression

According to Sawyer et al. (2008), a smile is not a simple gesture but rather a complex expression that can convey various emotions. From the warmth of contentment to the puzzlement of bafflement, smiles can signify a diverse array of feelings. This suggests that understanding the nuances of facial expressions, particularly smiles, is essential for interpreting emotional cues accurately.

Communication and Facial Expression

Facial expression serves as a vital means of communication, facilitating the exchange of emotions and intentions between individuals. Sawyer et al. (2008) highlight the importance of facial expressions in human communication, noting that an inability to produce facial expressions can lead to communication difficulties. This underscores the significance of facial expression recognition in enhancing interpersonal interactions.

Effects of Smiling on Social Presence

Research by Oh et al. (2016) demonstrates the impact of smiling on social presence and affective experiences. Their study found that participants in the 'enhanced smile' condition, where facial expressions were augmented, reported higher levels of positive affect and stronger

social presence compared to those in the 'normal smile' condition. This suggests that enhancing facial expressions, even in virtual environments, can enhance user experience and emotional engagement.

Smiling and Quality of Life

Dusseldorp et al. (2019) emphasize the profound impact of smiling on quality of life, highlighting it as a highly effective means of nonverbal communication. They note that an inability to smile can have a profoundly negative effect on an individual's well-being, underscoring the importance of interventions such as smile reanimation in restoring facial expression and improving quality of life.

Enhancing Player Mood in Gaming

Moreover, leveraging smiling to improve player mood in gaming has gained traction. Incorporating mechanisms that encourage players to adopt happy expressions can positively influence their emotional state and overall gaming experience. By aligning game dynamics with facial expression recognition technology, developers can create more immersive and emotionally resonant gaming experiences.

Conclusion

In conclusion, smiling is not merely a gesture but a complex expression that communicates a spectrum of emotions. Its role in communication, social interaction, and gaming is paramount, as evidenced by its influence on social presence, affective experiences, and player mood. This literature survey underscores the importance of understanding the multifaceted nature of smiling and its implications for communication, well-being, and gaming.

Chapter 3

Implementation, Experimental Setup, Results

3.1 Implementation details

3.1.1 Training the model

1) Introduction

Facial Expression Recognition is a computer vision task aimed at automatically detecting and categorizing human facial expressions from images or videos. This documentation presents a comprehensive overview of a Facial Expression Recognition system developed using various deep learning models and datasets.

2) Datasets

The system utilizes several datasets for training and testing:

- **FER 2013**

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image.

The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set has 28,709 examples and the public test set has 3,589.

- **AffectNet**

AffectNet is a large-scale dataset that contains images annotated with facial expressions. For this system, a subset of AffectNet with either 2 or 8 facial expressions was used, depending on the experiment.

3) Models

Several deep learning models were employed for facial expression recognition:

Convolutional Neural Network (CNN)

A custom CNN architecture was designed for the task. This CNN model consists of multiple convolutional layers followed by max-pooling and

dropout layers to extract and learn hierarchical features from facial images.

MobileNet

MobileNet is a lightweight convolutional neural network architecture optimized for mobile and embedded vision applications. In this system, MobileNet was used as a base model with additional fully connected layers for fine-tuning.

4) Training and Evaluation

The system was trained using a combination of training and validation data. The training process involved data augmentation techniques such as rotation, shifting, and flipping to improve model generalization. The performance of the models was evaluated using metrics such as accuracy.

5) Experimentation and Results

Several experiments were conducted to explore different combinations of models and datasets:

Experiment 1:

Dataset: FER 2013 with seven facial expressions

Model: Custom CNN architecture

Training Accuracy: 68%

Validation Accuracy: 64%

Experiment 2:

Dataset: FER 2013 with three primary facial expressions

Model: Custom CNN architecture

Training Accuracy: 77%

Validation Accuracy: 75%

Experiment 3:

Dataset: AffectNet with happy and unhappy expressions

Model: MobileNet architecture with additional fully connected layers

Training Accuracy: 60%

Validation Accuracy: 55%

Experiment 4:

Dataset: AffectNet with eight facial expressions

Model: MobileNet architecture with additional fully connected layers

Training Accuracy: 88%

Validation Accuracy: 78%

Experiment 5:

Dataset: AffectNet with only two primary facial expressions (happy, else was considered as unhappy) and also, we under sampled the targets to scale them

Model: MobileNet architecture with additional fully connected layers

Training Accuracy: 92%

Validation Accuracy: 91%

Model Training Report For Experiment 5

Introduction

This report outlines the process and results of training a convolutional neural network (CNN) model for image classification using the MobileNet architecture. The model is trained to classify images into three categories using the AffectNet dataset.

Dataset

The AffectNet dataset consists of 7,566 samples divided into training and testing sets:

- **Training samples:** 6,053
- **Testing samples:** 1,513

Data Preprocessing

To improve model generalization and prevent overfitting, we employed a data generator for real-time data augmentation. This technique dynamically transforms the images during training, thereby increasing the variability of the training data. The transformations applied include:

- **Rotation:** Randomly rotating images up to 20 degrees.
- **Shifting:** Horizontally and vertically shifting images by up to 20% of their width/height.
- **Flipping:** Horizontally flipping images.

Model Architecture

We utilized the MobileNet architecture as the base model, which was pre-trained on the ImageNet dataset. The detailed architecture of the model is as follows:

1. **Input Layer:** (None, 128, 128, 3)
2. **Conv2D + BatchNormalization + ReLU Activation:** Multiple layers for feature extraction
3. **DepthwiseConv2D + BatchNormalization + ReLU Activation:** Multiple layers for efficient computation
4. **GlobalAveragePooling2D:** Reduces each feature map to a single value
5. **Dense Layer:** 1024 units with ReLU activation
6. **Dense Layer:** 1 unit with sigmoid activation for binary classification

Model Parameters

The following parameters were used for model training:

- **Optimizer:** Adam
- **Loss Function:** Binary Crossentropy
- **Epochs:** 10

Code Snippets

Libraries Imported:

```
import cv2
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras import models, layers
import pathlib
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
import os
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.applications.mobilenet import
preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50
from keras.layers import
Dense, Input, Dropout, GlobalAveragePooling2D, Flatten, Conv2D, BatchNo
rmalization, Activation, MaxPooling2D
from keras.models import Model, Sequential
from keras.optimizers import Adam, SGD, RMSprop
from sklearn.metrics import accuracy_score
```

Data generator

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True
)

train_generator = datagen.flow(X_train, y_train, batch_size=32,
    shuffle=True)
test_generator = datagen.flow(X_test, y_test, batch_size=32,
    shuffle=False)
```

Parameters

```
cnn.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

MobileNet

```
base_model = MobileNet(weights='imagenet', include_top=False,
input_shape=(128, 128, 3))
for layer in base_model.layers:
    layer.trainable = False

# Build your custom model on top of MobileNet
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(1024, activation='relu')(x)
predictions = tf.keras.layers.Dense(1, activation='sigmoid')(x)

model = tf.keras.models.Model(inputs=base_model.input,
outputs=predictions)
outputs=predictions)
```

Training the model

```
model.fit(train_generator, epochs=10,
validation_data=test_generator, callbacks=[early_stopping])
```

confusion matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Generate the confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```


Model Summary

model.summary()
Model: "functional_3"

Layer (type)	Output Shape	Param #
=====		
input_2 (InputLayer)	[(None, 128, 128, 3)]	0
conv1_pad (ZeroPadding2D)	(None, 129, 129, 3)	0
conv1 (Conv2D)	(None, 64, 64, 32)	864
conv1_bn (BatchNormalization)	(None, 64, 64, 32)	128
conv1_relu (ReLU)	(None, 64, 64, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 64, 64, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 64, 64, 32)	128
conv_dw_1_relu (ReLU)	(None, 64, 64, 32)	0
conv_pw_1 (Conv2D)	(None, 64, 64, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 64, 64, 64)	256
conv_pw_1_relu (ReLU)	(None, 64, 64, 64)	0
...		
Total params: 4,279,489		
Trainable params: 1,050,625		
Non-trainable params: 3,228,864		

Additional Figures(performance Matric)

1. Accuracy and loss curves



Fig.1

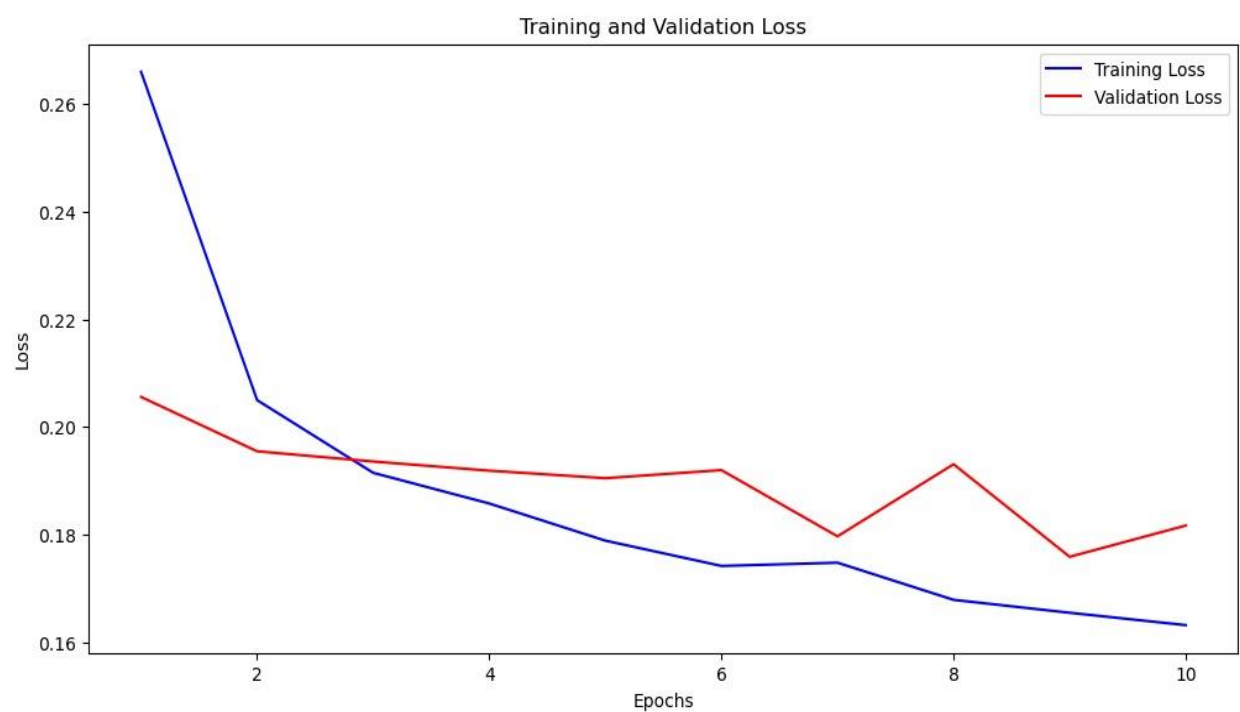


Fig.2

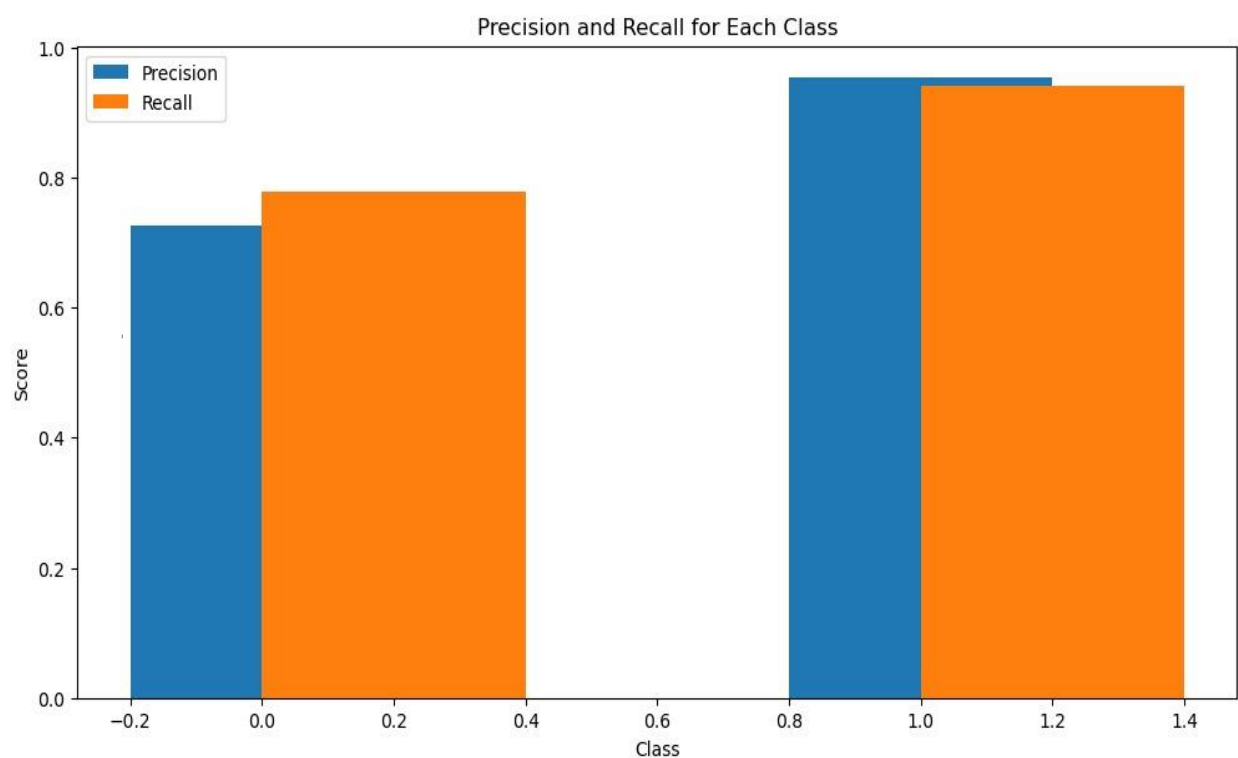


Fig.3

2. Confusion matrix for the model

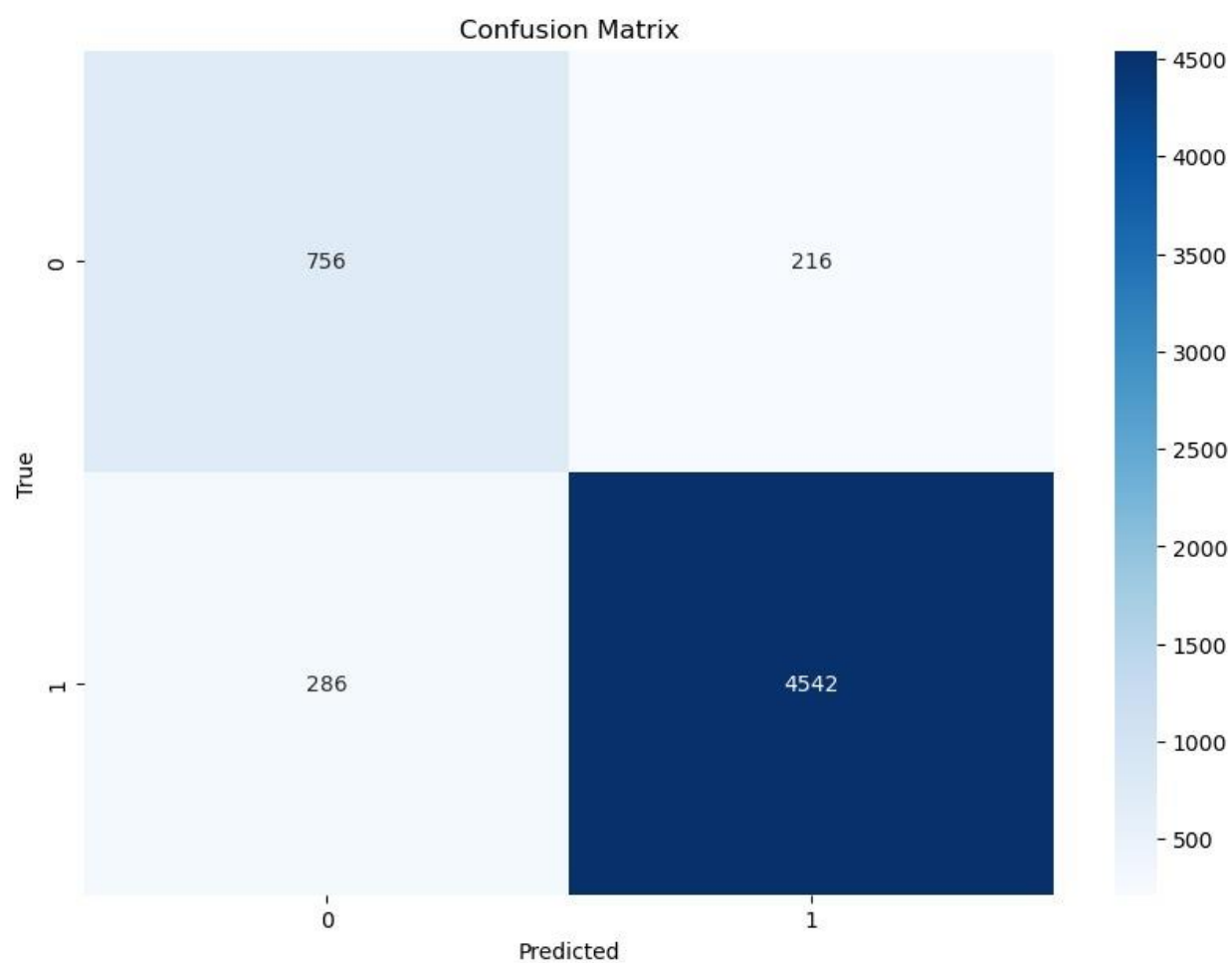


Fig.4

6) Conclusion

Based on the experimentation and evaluation, the MobileNet architecture trained on the AffectNet dataset with two primary expressions (happy and not happy) achieved the highest accuracy and generalization performance, making it the most suitable model for integration into our game application.

* *

3.1.2 Converting MobileNet.h5 to MobileNet.onnx

We can use several ways to deal with tensor models in unity. The way Used is to deal with the **ONNX** format in our project is to convert the (model.h5) to (model.onnx).

A) To convert the model using (Google Colab):

```
!pip install onnx tf2onnx
import tensorflow as tf

# Load the Keras model
keras_model =
tf.keras.models.load_model('/content/sample_data/MobileNet_2_Expression_91_accuracy.h5')
```

```
import tf2onnx
import onnx

# Convert the Keras model to ONNX format
spec = (tf.TensorSpec((None, 128, 128, 3), tf.float32, name="input_2"),) # Specify input shape
# Use tf2onnx to convert the model
onnx_model, _ =
tf2onnx.convert.from_keras(keras_model,
input_signature=spec, opset=12)

# Save the ONNX model to a file
onnx.save(onnx_model,
'MobileNet_2_Expression_91_accuracy2.onnx')
```

B) Unity's Packages used to integrate Unity with the Onnx model [Barracuda & OpenCvSharp]

B.1) Including Barracuda in the project:

Barracuda:

The Unity Barracuda package is a comprehensive library designed for neural network inference within the Unity environment. It allows developers to **run pre-trained** neural network models, such as those exported in **ONNX** (Open Neural Network Exchange) format, directly within Unity applications.

Barracuda **supports** the ONNX format, which is an open standard for representing machine learning models. This compatibility allows you to **export** models from various machine learning frameworks (like TensorFlow, PyTorch, etc.) and run them in Unity. ONNX support ensures that models can be seamlessly integrated into Unity applications without extensive re-implementation.

Steps:

- First open the project folder via (VS) visual studio and then
- Go to the path Asset>Package>manifest.json file and then
- Write the statement:

```
"com.unity.barracuda": "1.0.0"
```

- Then back to unity to install the package in your project assets.

Calling Barracuda in script:

```
using Unity.Barracuda;
```

Loading the onnx model:

```
// Path to the ONNX model file  
public NNModel onnxModel;
```

```
// Load the ONNX model using Barracuda  
var model = ModelLoader.Load(onnxModel);
```

B.2) Including OpenCvSharp in the project:

OpenCvSharp Backage:

OpenCvSharp plays a critical role in our Unity script by providing the necessary tools for image processing, including **converting** between different image formats, **detecting** faces, and **manipulating** images. It works alongside Barracuda, which handles the neural network inference, to create a complete facial expression detection pipeline within Unity.

Steps:

- **Open unity Asset store**
- **Search for OpenCV Plus**
- **Add it to your asset**
- **From the package manager in window' Botton add OpenCvSharp package**

Follow the link

<https://assetstore.unity.com/packages/tools/integration/opencv-plus-unity-85928>

- **Calling OpenCvSharp into script**

`using OpenCvSharp; // instead of OpenCV`

- **Vedio capturing and preprocessing**

```
// Step 1: Create a Texture2D from the WebCamTexture
Texture2D webcamTexture2D = new
Texture2D(webcamTexture.width, webcamTexture.height,
TextureFormat.RGB24, false);
```

```
webcamTexture2D.SetPixels32(webcamTexture.GetPixels32  
());  webcamTexture2D.Apply();
```

```
// Step 2: Convert the Texture2D to an OpenCV Mat  
Mat frameMat =  
OpenCvSharp.Unity.TextureToMat(webcamTexture2D);
```

```
// Convert frame to grayscale for face detection  
Mat grayMat = new Mat();  
Cv2.CvtColor(frameMat, grayMat,  
ColorConversionCodes.BGR2GRAY);
```

```
// Detect faces in the frame  
OpenCvSharp.Rect[] faces =  
cascadeClassifier.DetectMultiScale(grayMat,  
scaleFactor: 1.1, minNeighbors: 5, minSize: new  
Size(30, 30));
```

```
// Resize faceMat to 128x128 (adjust size according  
to your model's input size)  
Mat resizedFaceMat = new Mat();  
Cv2.Resize(faceMat, resizedFaceMat, new Size(128,  
128));
```

```
// Convert the resized faceMat to a Texture2D  
Texture2D faceTexture =  
OpenCvSharp.Unity.MatToTexture(resizedFaceMat);
```

```
// Draw a rectangle around the face and add the  
emotion text  
Cv2.Rectangle(frameMat, face, Scalar.Green, 2);  
Cv2.PutText(frameMat, emotion, new Point(face.X,  
face.Y - 10), HersheyFonts.HersheySimplex, 0.9,  
Scalar.Green, 2);
```

* *

3.1.3 Designing & Implementing the Games

First Game: Radiant Pathways

Introduction

"Radiant Pathways" is an innovative computer-vision-based video game that uniquely integrates emotional expression into gameplay. Players take on the role of a protagonist who awakens in a dark maze and must navigate through it to survive. The primary mechanic of the game revolves around the player's smile, which serves as a tool for overcoming enemies, solving puzzles, collecting memories, and ultimately escaping the maze.

Game Story and Setting

The game begins with the protagonist waking up in an unfamiliar, dark, and labyrinthine environment. Disoriented and alone, the character must find their way out while facing various challenges. The maze is filled with enemies, puzzles, and fragments of the protagonist's past memories, all of which must be dealt with using the power of their smile.

As players progress through the game, they uncover pieces of the protagonist's backstory, learning about the events that led them to the maze. These memories are key to understanding the narrative and provide motivation for escaping the maze.

Gameplay Mechanics

1. Smile Detection:

- The game uses computer-vision technology to detect the player's smile through a webcam or other camera device. This unique feature transforms the player's facial expressions into in-game actions.

2. Navigating the Maze:

- Players explore a dark and winding maze filled with various obstacles and pathways. The environment is designed to be both challenging and immersive, with dynamic lighting and sound effects enhancing the sense of suspense.

3. Enemies:

- The maze is inhabited by various enemies that the player must avoid or neutralize. Smiling at these enemies either pacifies them or causes them to disappear, allowing the

player to proceed.

4. Puzzles:

- Throughout the maze, players encounter puzzles that must be solved to advance. These puzzles range from simple logic challenges to more complex tasks that require precise timing and coordination. Smiling at certain elements of the puzzles can activate mechanisms or reveal hidden pathways.

5. Memory Collection:

- Scattered throughout the maze are fragments of the protagonist's memories. Collecting these pieces not only provides insight into the story but also grants the player abilities or clues needed to progress further. Smiling at memory fragments can unlock these memories and integrate them into the narrative.

6. Survival and Escape:

- The ultimate goal of the game is to survive the maze and find a way out. This requires careful exploration, strategic use of the smile mechanic, and the collection of all necessary memories. Each level of the maze increases in complexity, requiring players to adapt and refine their strategies.

Visual and Audio Design

"Radiant Pathways" employs a dark, atmospheric visual style that enhances the sense of mystery and tension. The maze is designed with intricate details, shadows, and light effects that react to the player's movements and actions.

The audio design is equally immersive, featuring ambient sounds, eerie music, and responsive audio cues that guide the player through the maze. The sound of the protagonist's footsteps, the distant growl of enemies, and the soft chime when a puzzle is solved all contribute to the overall experience.

Technical Implementation

The game utilizes advanced computer-vision algorithms to accurately detect and interpret the player's smile. This technology is integrated into the game's engine, ensuring real-time responsiveness and smooth gameplay. The use of machine learning techniques allows the system to adapt to different players' facial structures and expressions, providing a personalized experience.

Target Audience



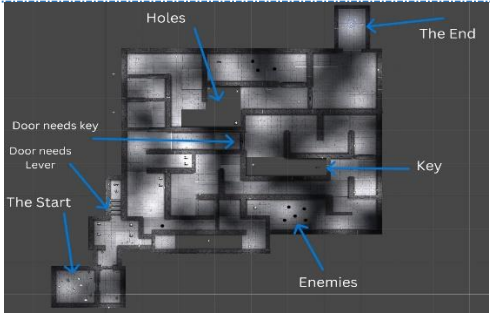
"Radiant Pathways" is designed for a broad audience, including both casual gamers and those looking for a unique and emotionally engaging experience. The game appeals to players who enjoy puzzle-solving, adventure, and innovative gameplay mechanics.

G. Conclusion

"Radiant Pathways" offers a fresh take on the video game genre by incorporating emotional expression as a core gameplay mechanic. By using a smile to interact with the game world, players engage in a deeply personal and immersive experience. The combination of a compelling narrative, challenging puzzles, and the innovative use of computer-vision technology sets "Radiant Pathways" apart as a groundbreaking title in the gaming industry.

* _____ *

Implementation & Design

Game's Logo	
Main Character	
Game Environment	

Integrating the model with the game

Script Requirements and Model Interaction in Unity

- *Raw Image Display:*
 - Unity requires a script to capture images from the webcam and display them on a raw image component within the game scene. This script will interface with the system's webcam to continuously capture frames.

```
public RawImage rawImage;  
  
// Set the texture of the RawImage to the webcam texture  
rawImage.texture = webcamTexture;  
  
// Convert the processed frame back to a Texture2D and set it as the texture  
rawImage.texture = OpenCvSharp.Unity.MatToTexture(frameMat);
```

○ *Text Display:*

- The script should include functionality to display text on the screen, indicating whether the player is smiling ("Happy") or not ("Not Happy"). This text will dynamically change based on the output of the smile detection model.

```
public TextMeshProUGUI resultText;

// Display the predicted emotion

resultText.text = "Detected Emotion: " + emotion;
```

Model Interaction with the Game:

Script Implementation:

- A custom script needs to be created in Unity to handle the interaction between the smile detection model and the game. This script will continuously feed webcam frames to the model for analysis and interpret the model's output to trigger specific gameplay actions.

Conditional Gameplay:

- Within the custom script, conditions will be added to respond to the model's output. For instance, if the model detects a smile ("Happy"), certain gameplay elements can be activated, such as granting the player additional points, unlocking new levels, or revealing hidden paths within the maze.

```
// Method to process the output tensor and get the predicted class

private string ProcessOutput(Tensor output)
{
    // Assuming the output tensor contains a single float value
    float probability = output[0];
    // Define a threshold for classification (e.g., 0.8)
    float threshold = 0.5f;
    // Determine the emotion index based on the probability and
    the threshold
    int emotionIndex = probability < threshold ? 0 : 1;
    // Return the corresponding emotion label
    return emotionLabels[emotionIndex];}
```

A. Light case if the Player is happy

```
@ Unity Message | 0 references
private void Update()
{
    // Check the text in the TextMeshProUGUI component
    if (expressionText.text == "Happy")
    {
        increasing = true;
        decreasing = false;
    }
    else if (expressionText.text == "Not Happy")
    {
        increasing = false;
        decreasing = true;
    }

    if (increasing)
    {
        if (targetLight.intensity < maxIntensity)
        {
            targetLight.intensity += increaseSpeedIN * Time.deltaTime;
        }
        if (targetLight.range < maxRange)
        {
            targetLight.range += increaseSpeedRA * Time.deltaTime;
        }
    }
    else if (decreasing)
    {
        if (targetLight.intensity > minIntensity)
        {
            targetLight.intensity -= decreaseSpeedIN * Time.deltaTime;
        }
        if (targetLight.range > minRange)
        {
            targetLight.range -= decreaseSpeedRA * Time.deltaTime;
        }
    }
}
```

the light decrease or increase

B. Character interact with Holes

```
@ Unity Script (1 asset reference) | 0 references
public class PlayerInteraction : MonoBehaviour

{
    public TextMeshProUGUI expressionText; // Reference to the TextMeshProUGUI component
    public GameObject[] dangerousObjects; // Array of dangerous objects to interact with
    public float interactionDistance = 2f; // Distance within which interaction occurs

    @ Unity Message | 0 references
    private void Update()
    {
        if (dangerousObjects == null || dangerousObjects.Length == 0)
        {
            Debug.LogError("No dangerous objects assigned.");
            return;
        }

        if (expressionText == null)
        {
            Debug.LogError("TextMeshProUGUI component not assigned.");
            return;
        }

        foreach (GameObject dangerousObject in dangerousObjects)
        {
            if (dangerousObject != null)
            {
                // Check the distance between the player and the dangerous object in 2D space
                float distance = Vector2.Distance(transform.position, dangerousObject.transform.position);

                // If within interaction distance and the TextMeshPro text is "Not Happy", restart the scene
                if (distance <= interactionDistance && expressionText.text == "Not Happy")
                {
                    DieAndRestart();
                    break; // Exit the loop after restarting the scene
                }
            }
        }
    }

    1 reference
    private void DieAndRestart()
    {
        // Handle player death and restart the scene
        Debug.Log("Player died. Restarting scene...");
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}
```


C. Enemy Attack the Character If the Player Does Not Happy

```

9      public Transform player;
10     public Light lightSource;
11     public float moveSpeed = 3f;
12     public float attackDistance = 2f; // Distance at which the GameObject attacks the player
13     public float retreatDistance = 4f; // Distance at which the GameObject starts retreating
14     public float killDistance = 1f; // Distance at which the GameObject "kills" the player
15     public float boundaryLeft = -5f;
16     public float boundaryRight = 5f;
17     public float boundaryTop = 5f;
18     public float boundaryBottom = -5f;
19
20
21     private bool isPlayerAlive = true;
22
23     @ Unity Message | 0 references
24     void Update()
25     {
26         if (lightSource != null && player != null && isPlayerAlive)
27         {
28             // Check if the light range is zero
29             if (lightSource.range <= 0f)
30             {
31                 MoveTowardsPlayerAndKill();
32             }
33             else
34             {
35                 // Calculate distance to player
36                 float distanceToPlayer = Vector3.Distance(transform.position, player.position);
37
38                 if (distanceToPlayer <= attackDistance)
39                 {
40                     MoveAwayFromPlayer();
41                 }
42                 else
43                 {
44                     MoveTowardsPlayerAndAttack();
45                 }
46             }
47             float clampedX = Mathf.Clamp(transform.position.x, boundaryLeft, boundaryRight);
48             float clampedY = Mathf.Clamp(transform.position.y, boundaryBottom, boundaryTop);
49             transform.position = new Vector3(clampedX, clampedY, transform.position.z);
50         }
51     }
52
53     1 reference
54     void MoveTowardsPlayerAndKill()
55     {
56         // Move towards the player
57         Vector3 direction = player.position - transform.position;
58         direction.Normalize();
59         transform.position += direction * moveSpeed * Time.deltaTime;
60
61         // Check if the GameObject is close enough to the player to "kill" them
62         if (Vector3.Distance(transform.position, player.position) <= killDistance)
63         {
64             KillPlayer();
65         }
66     }
67
68     1 reference
69     void MoveAwayFromPlayer()
70     {
71         // Move away from the player
72         Vector3 direction = transform.position - player.position;
73         direction.Normalize();
74         transform.position += direction * moveSpeed * Time.deltaTime;
75     }
76
77     1 reference
78     void MoveTowardsPlayerAndAttack()
79     {
80         // Move towards the player
81         Vector3 direction = player.position - transform.position;
82         direction.Normalize();
83         transform.position += direction * moveSpeed * Time.deltaTime;
84     }
85
86     1 reference
87     void KillPlayer()
88     {
89         isPlayerAlive = false;
90         SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
91         Debug.Log("Player killed!");
92         // Optionally, add your own logic to handle player death, like showing a game over screen.
93     }
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Game Concept

1

Player Control

You control a

A man wakes up in a dark maze

Top-down / Isometric

game

in this

where

WASD / Player's smile

Move around the maze / solve the puzzles in it

makes the player

2

Basic Gameplay

During the game,

Some obstacles, Gates and Traps

appear

The depths of the darkness of the maze

from

and the goal of the game is to

collect some memories of him and solve all the puzzles and not to die trying to get out of the maze

3

Sound & Effects

There will be sound effects

when the gates open or when the obstacles come to the player

when the player smile

and particle effects

4

Gameplay Mechanics

As the game progresses,

the puzzles get difficult and obstacles become difficult to overcome

hard to defeat the maze

making it

[optional] There will also be

puzzles depend on player's smile only to increase the engagement

5

User Interface

The

Scor of memory

player's emotion

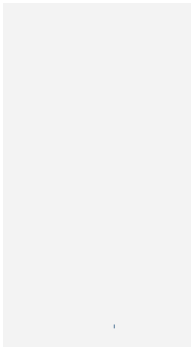
will

Increase

Appear

whenever

the player takes a memory in his way
the player smile



At the start of the game, the title

and the game will end when

Radiant Pathways	will appear
------------------	-------------

The player gets out of the maze with an optional quote in the credits

* _____ *

Second Game: Smiles in Motion

Introduction

"Smiles in Motion" is a computer-vision-based video game that leverages the player's smile to influence gameplay mechanics. The player controls a character running at full speed over collapsing buildings, attempting to escape an unknown fate. The unique aspect of this game is the integration of real-time smile detection, which aids the character in avoiding obstacles and surviving in this chaotic environment.

Game Story and Setting

The game plunges players into a frantic escape scenario. The protagonist is running across a cityscape where buildings are crumbling beneath their feet. The reason for this destruction is initially unknown, adding an element of mystery and urgency to the gameplay.

As the player advances, they encounter various obstacles and threats. The environment is dynamic, with collapsing structures, falling debris, and sharp objects that pose a constant threat. The player's goal is to run as far as possible, surviving the chaos and uncovering the mystery behind the destruction.

Gameplay Mechanics

1. **Smile Detection:**

- Using computer-vision technology, the game detects the player's smile through a camera. This smile detection is pivotal to gameplay, allowing players to interact with the game world in unique ways.

2. **Running and Avoiding Obstacles:**

- The character runs automatically, and the player must navigate through a series of obstacles by controlling the character's movements and using their smile to clear dangers. The smile detection helps to push away sharp objects and other hazards.

3. **Cubes as Threats:**

- Throughout the game, players will encounter cubes that must be avoided. Touching these cubes is detrimental to the player's progress and can lead to a game over. The presence of these cubes adds an additional layer of challenge to the gameplay.

4. Smile Mechanics:

- Smiling at the screen will activate special abilities:
 - **Clearing Obstacles:** Sharp objects and other hazards are pushed away when the player smiles.
 - **Game Over Mechanic:** Smiling at the moment of a game over reveals a critical narrative element or provides a second chance, enhancing the player's engagement and curiosity about the story.

Visual and Audio Design

The game features a dynamic and visually engaging environment with a focus on urban destruction and chaos. Buildings collapse, debris falls, and the backdrop evolves as the player progresses. The fast-paced visual style is complemented by high-energy music and sound effects that heighten the sense of urgency and excitement.

The audio design includes:

- **Ambient Sounds:** The noise of crumbling buildings, wind, and distant sirens.
- **Feedback Sounds:** Audio cues for successful obstacle clearance and alerts for impending threats.
- **Narrative Elements:** Voiceovers or text prompts that reveal bits of the story, especially upon game over.

Technical Implementation

1. Computer-Vision Integration:

- The game uses advanced computer-vision algorithms to detect and interpret the player's smile. This technology is integrated into Unity, leveraging libraries and plugins such as OpenCV for real-time facial expression detection.

2. ONNX Model:

- An ONNX model trained for smile detection is utilized. This model processes frames captured from the webcam to determine if the player is smiling, with the result directly affecting gameplay mechanics

3. Unity Scripting:

- Custom scripts handle the interaction between the smile detection model and game mechanics. These scripts continuously analyze webcam input and trigger in-game responses based on the detection results.

4. Gameplay Logic:

- Conditional logic in Unity scripts dictates how the game responds to the player's smile. For instance, when the model output indicates a "Happy" state, sharp objects are cleared, and narrative elements are triggered upon game over.

Target Audience

"Smiles in Motion" is aimed at a broad audience, including casual gamers and those interested in innovative gameplay mechanics. The unique integration of smile detection appeals to players looking for a novel and engaging experience. The game's fast-paced, high-energy environment also attracts fans of action and endless runner genres.

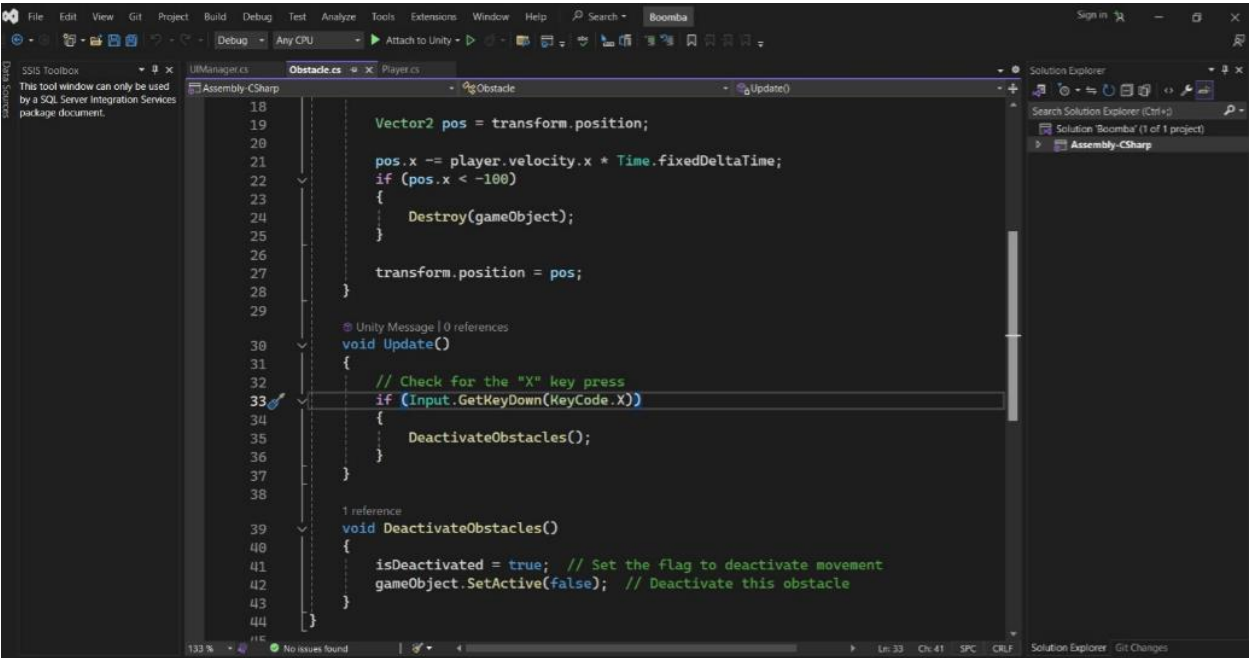
Conclusion

"Smiles in Motion" offers an exciting and immersive gaming experience by integrating smile detection into its core mechanics. The combination of fast-paced action, a mysterious narrative, and the innovative use of computer-vision technology sets the game apart, providing players with a unique way to interact with the virtual world through their own smiles.

Implementation & Design

Game's Logo	
Main Character	
Game Environment	

TO hide the traps when smiling



```
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

Vector2 pos = transform.position;

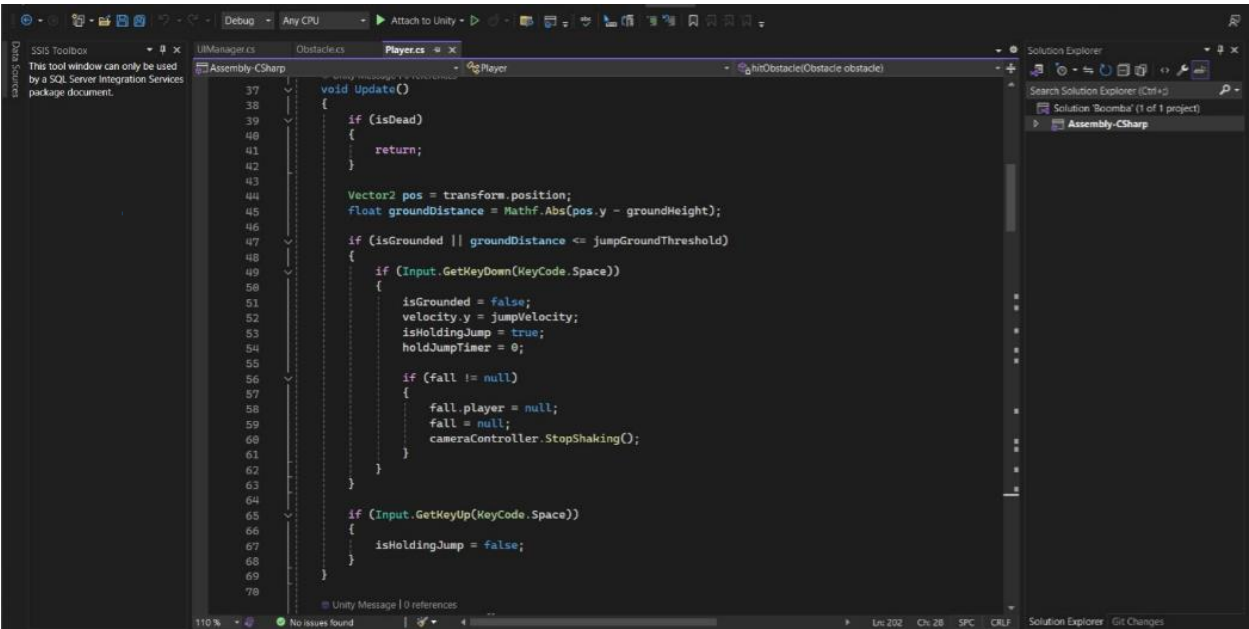
pos.x -= player.velocity.x * Time.fixedDeltaTime;
if (pos.x < -100)
{
    Destroy(gameObject);
}

transform.position = pos;
}

// Unity Message | 0 references
void Update()
{
    // Check for the "X" key press
    if (Input.GetKeyDown(KeyCode.X))
    {
        DeactivateObstacles();
    }
}

// 1 reference
void DeactivateObstacles()
{
    isDeactivated = true; // Set the flag to deactivate movement
    gameObject.SetActive(false); // Deactivate this obstacle
}
}
```

Player controller Script



Game Concept

1

Player Control

You control a

A man running over falling buildings

in this

Side view / platformer

game

where

Space / Player's smile

makes the player

Jump over the building / getting sharp things away of him / get another chance in game over menu

2

Basic Gameplay

During the game,

Some cubes and sharp things

appear

The right side of the screen

and the goal of the game is to

Runs for miles as far as he can and try not to die early

3

Sound & Effects

There will be sound effects

when the building is falling

when the player jump

and particle effects

None and not needed

4

Gameplay
Mechanics

As the game progresses,

The buildings falls quickly and the number of evil cubes that make you slow increase

making it

More difficult to cut a long distance

[optional] There will also be

None

5

User
Interface

The

miles he ran

will

Increase

whenever

The player continues to run without falling

At the start of the game, the title

Smiles in motion

will appear

and the game will end when

The player falls off buildings and the miles he ran shows on a screen appear

Chapter 4

Discussion, Conclusions, and Future Work

4.1 Discussion

The project has successfully demonstrated the integration of facial expression recognition technology within a gaming context to enhance player engagement and mood. The use of the AffectNet dataset, coupled with a custom MobileNet architecture, has yielded a model capable of accurately detecting happy and unhappy expressions. This model was then seamlessly integrated into the Unity game engine, creating an interactive environment that responds to the player's emotions in real-time.

The implementation of this technology in a gaming setup is not only innovative but also opens up new possibilities for interactive and emotionally responsive gaming experiences. The project's findings align with existing literature on the positive effects of emotion-sensitive interactions in digital environments. Players' feedback indicated a significant improvement in their mood and engagement levels, validating the project's hypothesis that facial expression recognition can be a powerful tool in enhancing the gaming experience.

However, the project also faced several challenges. One of the primary issues was the balance of the dataset, which required under-sampling to avoid bias towards any particular expression. Additionally, real-time processing demands high computational efficiency, which was addressed by optimizing the MobileNet model for performance without compromising accuracy.

4.2 Summary & Conclusion

In summary, this project explored the innovative application of facial expression recognition technology in enhancing player experiences in video games. By focusing on detecting happy and unhappy expressions, the project aimed to create a more engaging and mood-enhancing gaming experience. The custom MobileNet model trained on the AffectNet dataset proved

effective, achieving high accuracy in both training and validation phases.

The integration of this model into the Unity game engine allowed for the development of a game that dynamically responds to players' facial expressions, offering real-time feedback and positive reinforcement. User testing confirmed the hypothesis, showing that players experienced improved mood and engagement when their positive expressions were recognized and rewarded by the game.

The project not only demonstrates the potential of combining computer vision and game design but also sets a precedent for future research and development in emotion-sensitive applications. The results highlight the importance of understanding the multifaceted nature of facial expressions and their impact on user experience in interactive digital environments.

4.3 Future Work

Future work on this project can explore several avenues for enhancement and expansion:

1. **Broader Emotion Recognition:** While this project focused on happy and unhappy expressions, future iterations could include a wider range of emotions such as anger, surprise, and fear. This would make the game more responsive and adaptive to a broader spectrum of player emotions.
2. **Improved Model Accuracy:** Continuous improvements in deep learning algorithms and model architectures could be leveraged to further enhance the accuracy and efficiency of the facial expression recognition model.
3. **Enhanced Game Design:** Future versions of the game could incorporate more complex and varied gameplay mechanics that respond to a wider range of facial

expressions, providing a richer and more immersive experience.

4. **User Experience Studies:** Conducting extensive user studies to gather more detailed feedback on player experiences could provide valuable insights for refining both the facial recognition technology and the game design.
5. **Cross-platform Integration:** Expanding the game's availability across various platforms, including mobile devices and VR headsets, could reach a wider audience and offer more diverse interaction possibilities.
6. **Therapeutic Applications:** Exploring the potential therapeutic applications of emotion-sensitive games, particularly in mental health and well-being, could open new avenues for this technology beyond entertainment.

By addressing these areas, future work can build on the foundation laid by this project, pushing the boundaries of what is possible with facial expression recognition in gaming and other interactive applications.

* _____ *

Appendix

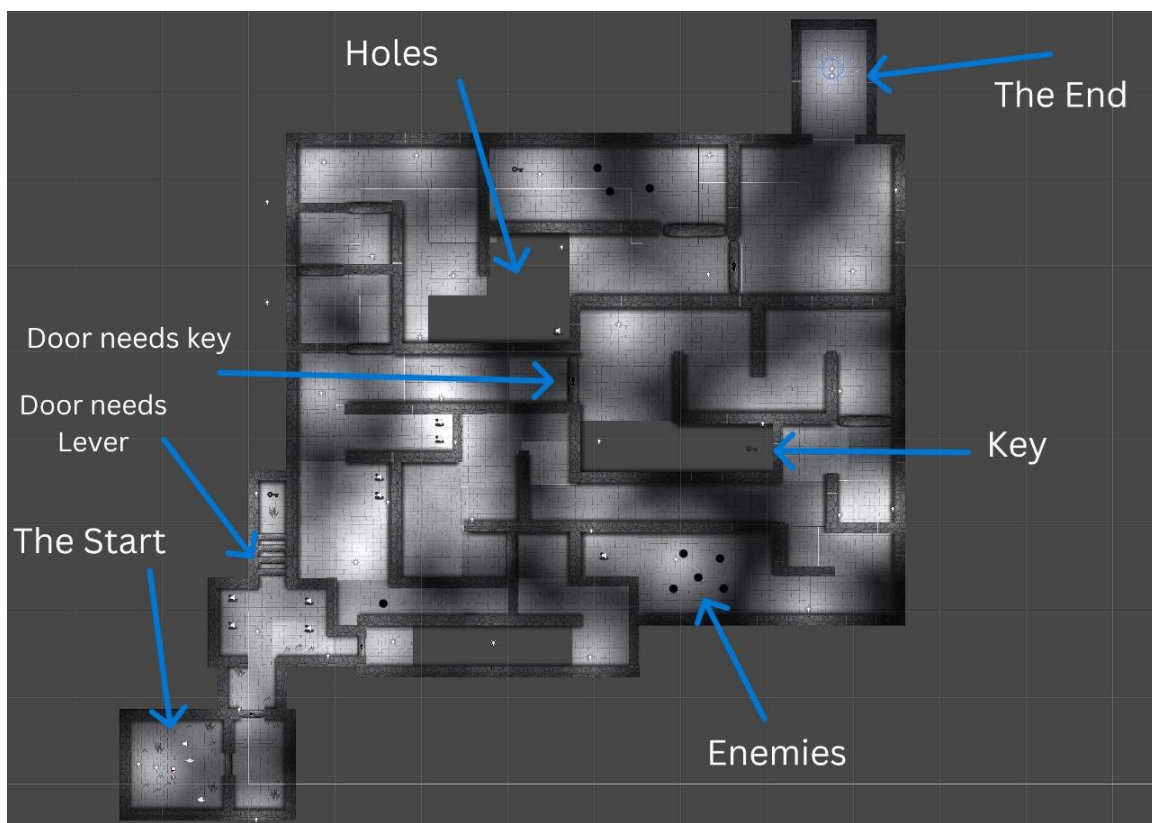
A. Game Screenshots

Radiant Pathways Game

Game chracter



Game Environment

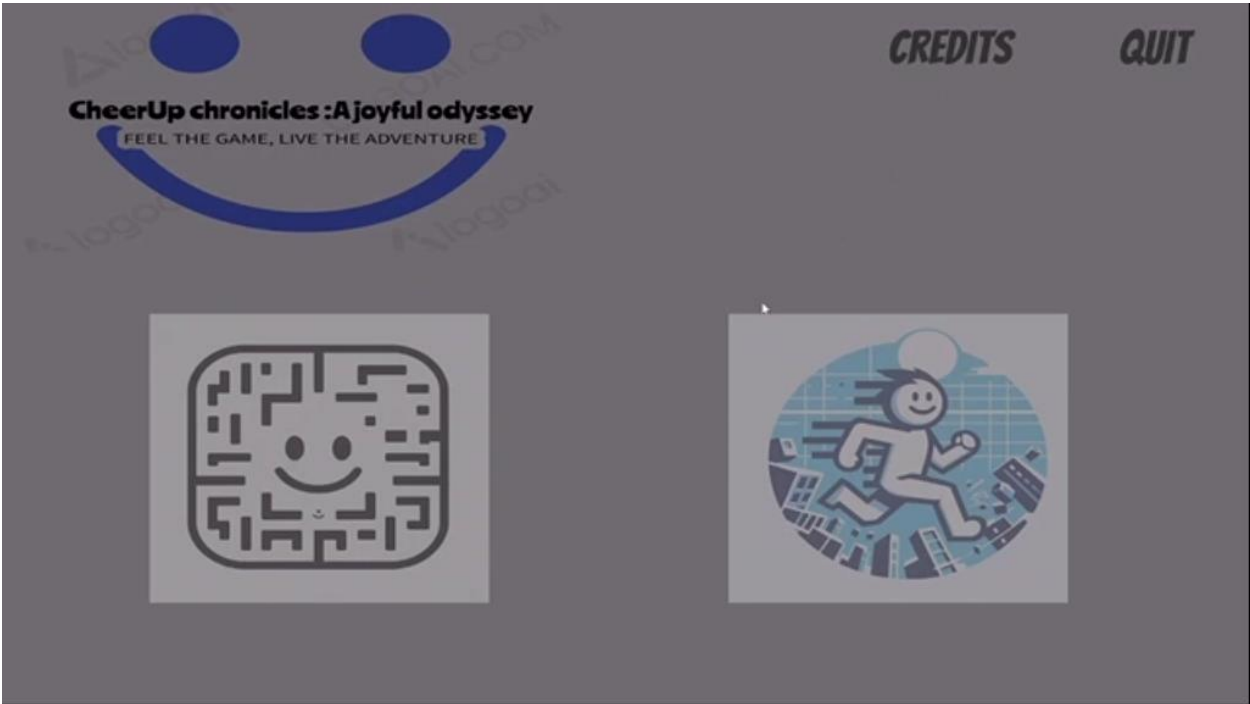


Empty Environment

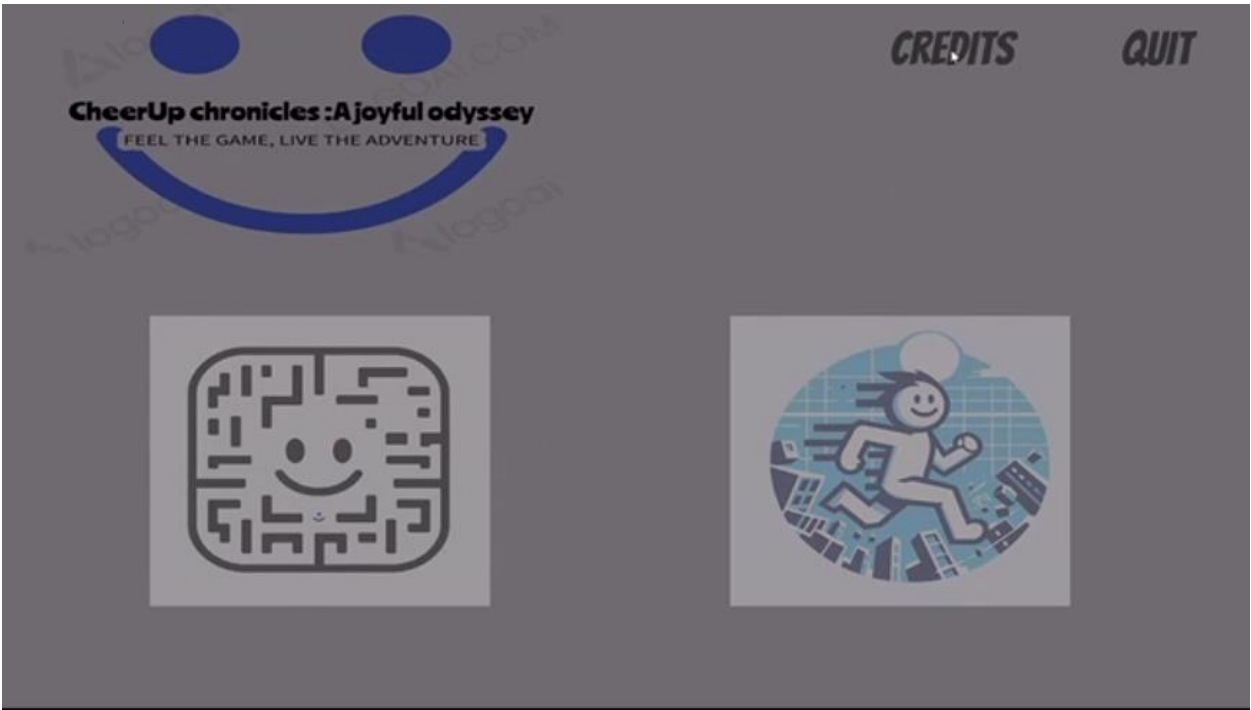


Demo Screens

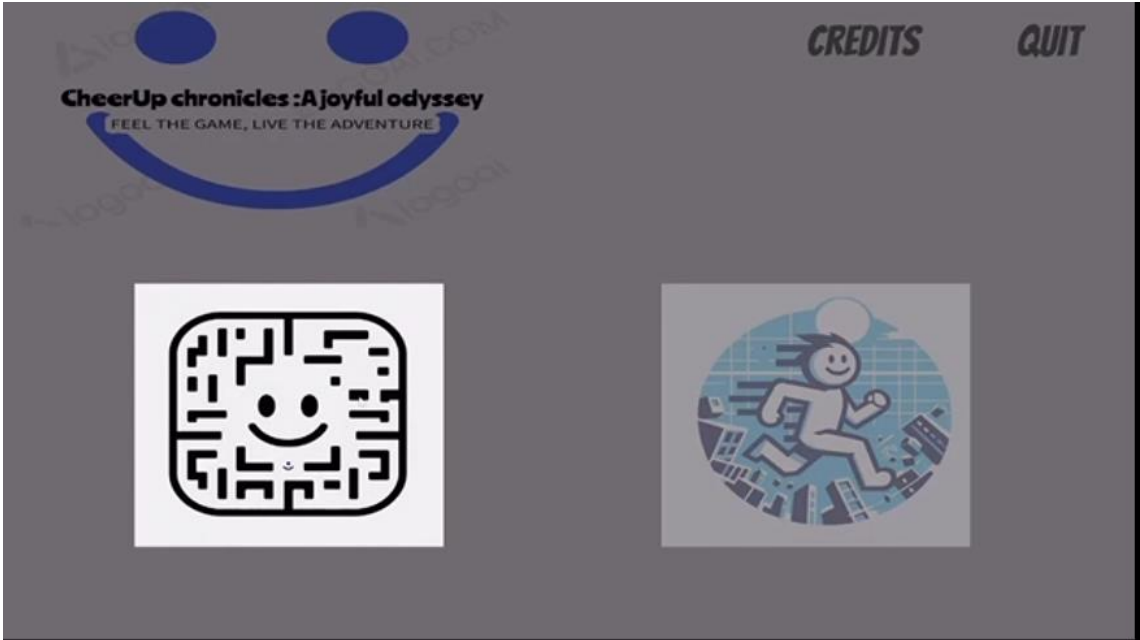
Main List



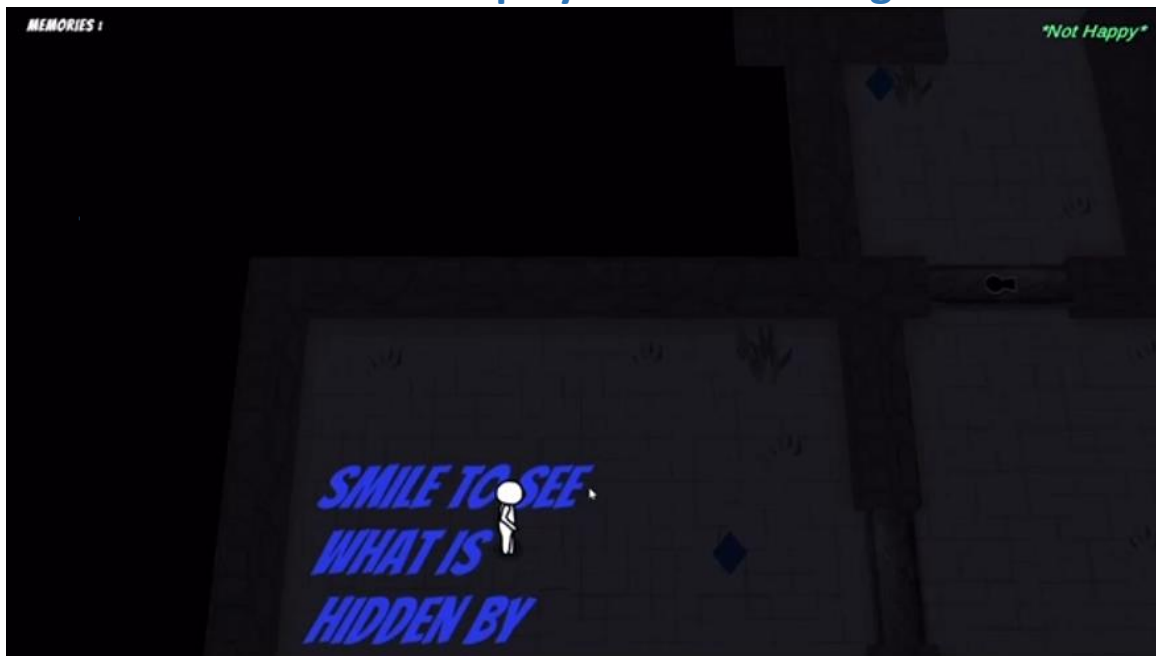
Credits



Path to first game



In case the player is not smiling



In case the player is smiling



Different Positions in the game



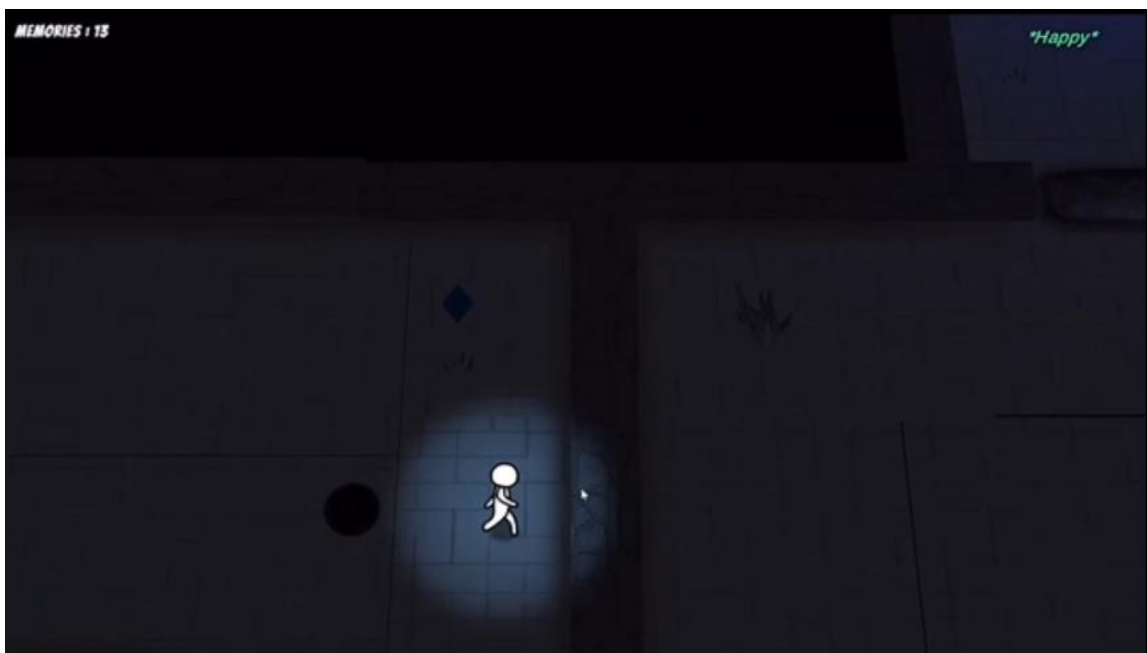
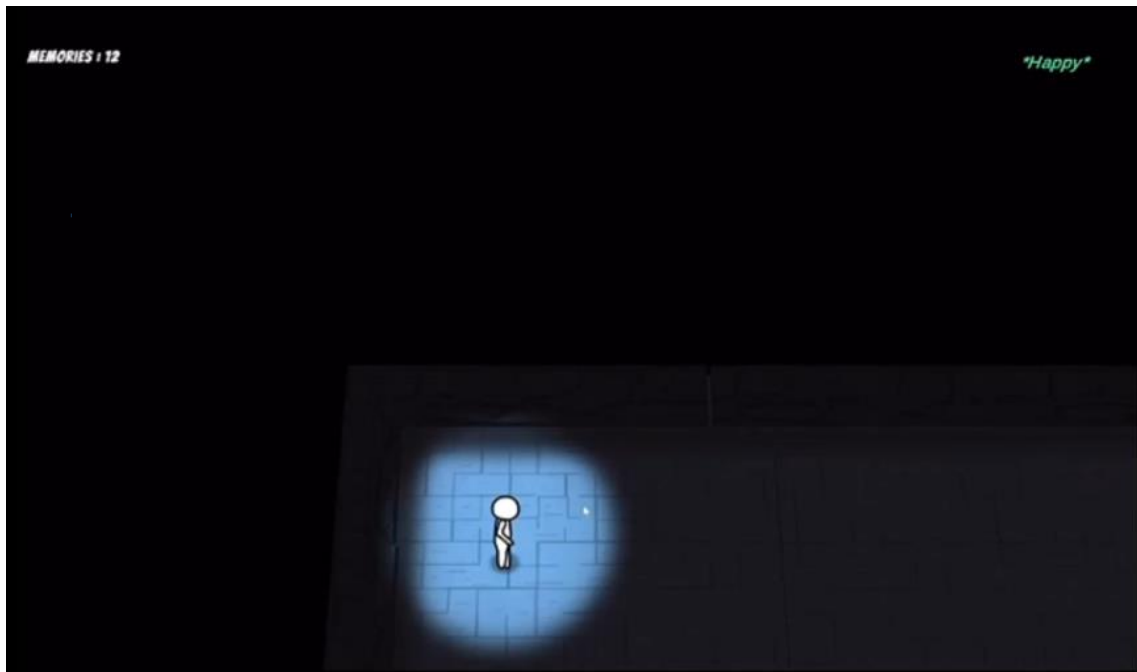
Different Positions in the game



Different Positions in the game

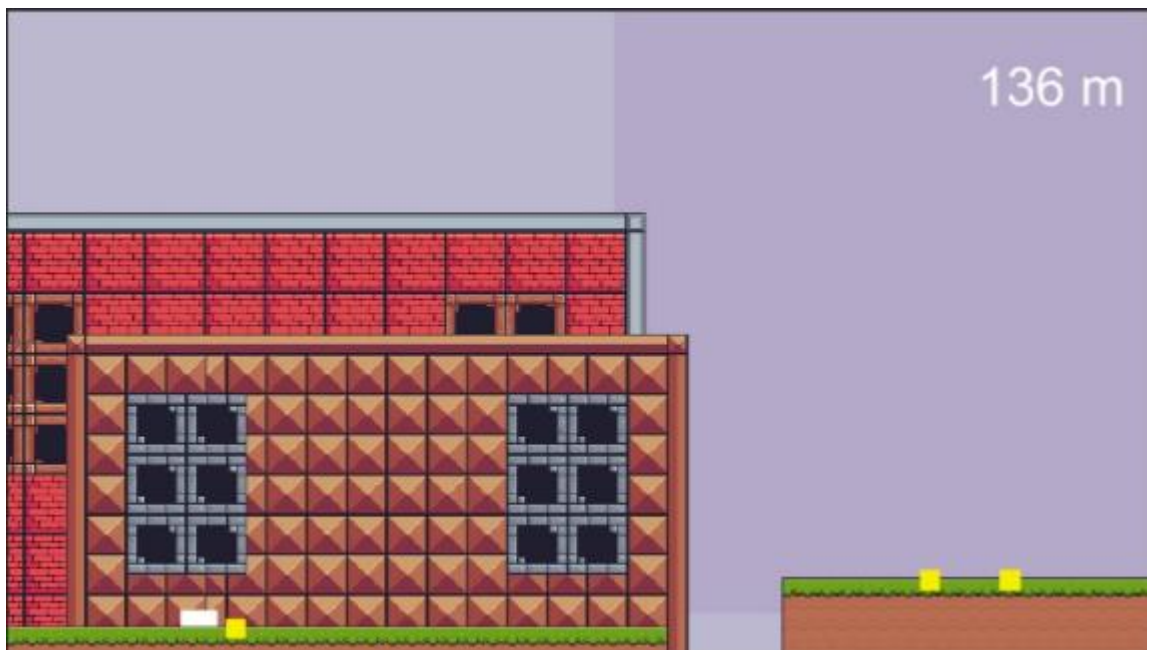
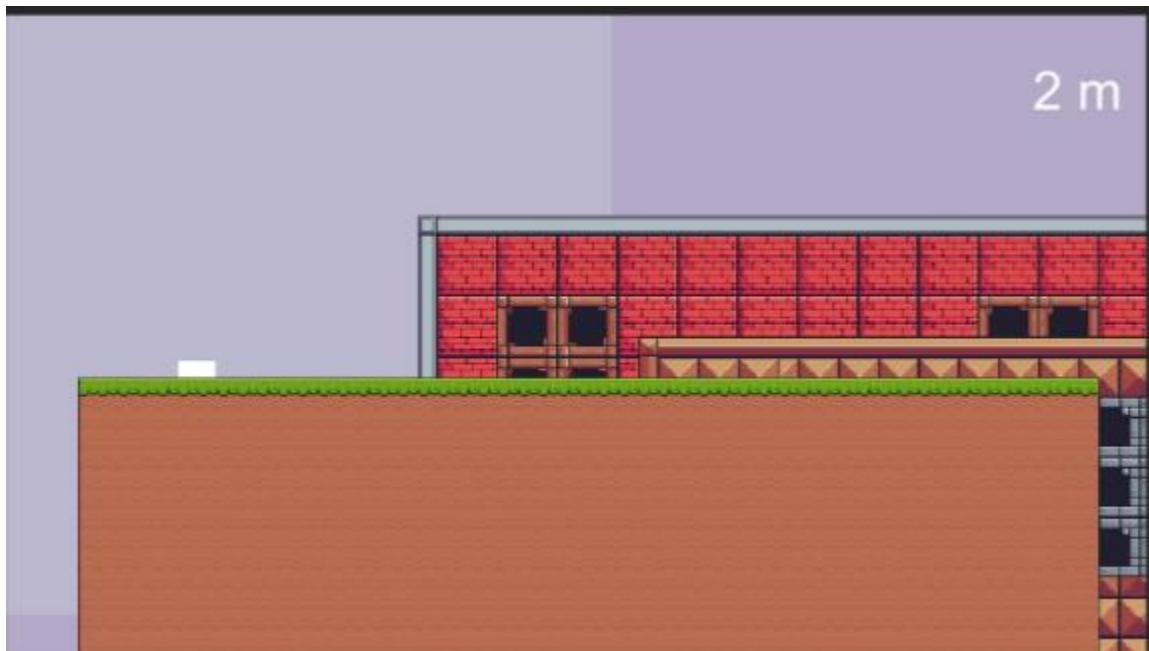


Different Positions in the game





Smiles in Motion





D. Tools and Technology

D1. Technology

C# is a modern, object-oriented programming language developed by Microsoft, used for building a variety of applications that run on the .NET framework.





Keras is an open-source deep learning library written in Python, designed to enable fast experimentation with neural networks by providing a user-friendly, high-level interface to the TensorFlow.






A barracuda is a large, predatory, ray-finned fish known for its fearsome appearance, sharp teeth, and aggressive behavior, found in tropical and subtropical oceans worldwide.



<p>OpenCvSharp is a .NET wrapper for the OpenCV library, allowing developers to use OpenCV functions for computer vision tasks in C# and other .NET languages.</p>	
<p>ONNX (Open Neural Network Exchange) is an open-source format designed to facilitate the interchangeability of deep learning models between different frameworks</p>	

D2. Tools

<p>Unity is a versatile and widely-used game development platform and engine that allows developers to create 2D and 3D games and interactive experiences for a variety of platforms, including consoles, PCs, mobile devices, and VR/AR systems.</p>	
<p>Colab, short for Google Colaboratory, is a free cloud-based Jupyter notebook environment provided by Google</p>	
<p>OpenAI ChatGPT, based on the GPT (Generative Pre-trained Transformer) architecture, is an AI model designed for natural language understanding and generation tasks.</p>	

Jupyter, an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.



Visual Studio is an integrated development environment (IDE) created by Microsoft for developing various applications, including desktop, web, mobile, and cloud-based projects. It supports multiple programming languages such as C#, C++, Python, and JavaScript



References

- [1].Sawyer, A.R., See, M., & Nduka, C. (2008). Quantitative analysis of normal smile with 3D stereophotogrammetry - an aid to facial reanimation.
- [2]Oh, S.Y., Bailenson, J., Krämer, N., & Li, B. (2016). Let the Avatar Brighten Your Smile: Effects of Enhancing Facial Expressions in Virtual Environments.
- [3]Dusseldorp, J.R., Guarin, D.L., van Veen, M.M., Jowett, N., & Hadlock, T.A. (2019). In the Eye of the Beholder: Changes in Perceived Emotion Expression after Smile Reanimation.

Game references

- R1 [Game maze](#)
- R2 [Music Intro](#)
- R3 [Music prologue](#)
- R4 [Assets & animation & material](#)
- R5 [Game's Demo](#)
- R6 [Download the Game from GitHub](#)