EX-4 Gaming Application with 2D-Animations and Gestures

Aim

To develop a gaming application with 2d-animations and gestures in flutter.

Definitions

Flutter

Flutter is not a programming language. It's a software development kit (SDK) with prewritten code, consisting of ready-to-use and customizable widgets, as well as libraries, tools, and documentation that together serve to build cross-platform apps.

Flutter plugin

A Flutter plugin is a special kind of package that enables Flutter apps to interact with platform-specific APIs (iOS, Android, web, desktop). Plugins can include Dart code, but crucially, they also contain platform-specific implementation code written in Kotlin/Java for Android and Swift/Obj-C for iOS.

Dart plugin

The Dart plugin adds Dart support to IntelliJ Platform-based IDEs developed by JetBrains. These IDEs provide features unique to specific development technologies. The IDEs recommended for Dart and Flutter development include: IntelliJ IDEA which specializes in JVM-based language development.

Flutter SDK

Flutter is Google's free, open-source software development kit (SDK) for cross-platform mobile application development. Using a single platform-agnostic codebase, Flutter helps developers build high-performance, scalable applications with attractive and functional user interfaces for Android or IOS.

Gaming Application

A gaming application is a web, mobile, or desktop application that enables users to play games against each other. Gaming apps may also include features for managing the game and interacting with friends. Games are often categorized as either casual games or immersive, engaging games.

2D-Animations

2D animation is the art of creating movement in a two-dimensional space. This includes characters, creatures, FX, and backgrounds. The illusion of movement is created when individual drawings are sequenced together over time. One second of time is usually divided into 24 frames.

Gestures

Gestures help users to navigate between views, take actions, and manipulate content. Types of gestures include: Navigational gestures and Action gestures.

Procedure

1. **Open android studio**
2. **Click ‘new flutter project’**
3. **Select ‘flutter’ at the left side of the window**
4. **Add ‘flutter sdk’ from the desired location**
5. **Click ‘next’ and specify the project name and select language ‘java’, check only Android, Web and Windows under platforms then click ‘create’**
6. **Create a new dart file under ‘lib’ folder in the projects window (right click over lib folder - > new -> dart file -> specify the file name as ‘game2d’ -> press ‘enter’**
7. **Type the following codes in the game2d.dart file**

Game2d.dart

```
import 'dart:math';

import 'package:flutter/material.dart';
void main() {
  runApp(
    MyApp(),
  );
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.green, // Set the app's primary theme color
      ),
      debugShowCheckedModeBanner: false,
      home: Bounceball(),
    );
  }
}
class Bounceball extends StatefulWidget {
  const Bounceball({Key? key}) : super(key: key);

  @override
  _BounceballState createState() => _BounceballState();
}

class _BounceballState extends State<Bounceball> with SingleTickerProviderStateMixin{
  bool isClick = false;
  bool isClickAfter = true;
  bool collapse = false;
```

```
var ball = myBall.origin();
late AnimationController _animationController;
double baseTime = 0.016;
double accel = 1000;

@override
void initState() {
  // TODO: implement initState
  super.initState();
  _animationController = AnimationController(
      vsync: this,
      duration: Duration(milliseconds: 1)
  );
  _animationController.repeat();
}

@override
void dispose(){
  _animationController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text("2D Bounce Ball Animation with Gestures"),
      ),
      body: Center(
          child: GestureDetector(
            onHorizontalDragDown: (details) {
              setState(() {
                if (ball.isBallRegion(details.localPosition.dx, details.localPosition.dy)) {
                  isClick=true;
                  ball.stop();
                }
              });
            },
            onHorizontalDragEnd: (details) {
              if (isClick) {
                setState(() {
                  isClick = false;
                  isClickAfter = true;
                });
              }
            },
```

```
            onHorizontalDragUpdate: (details) {
              if (isClick) {
                setState(() {
                  ball.setPosition(details.localPosition.dx, details.localPosition.dy);
                  ball.updateDraw();
                });
              }
            },

            child: AnimatedBuilder(
                animation: _animationController,
                builder: (context, child) {
                  if (!isClick) {
                    if (ball.yVel!=0 || isClickAfter) {
                      ball.addYvel(baseTime * accel);
                      ball.subYpos(0.5 * accel * pow(baseTime, 2) - ball.yVel * baseTime);
                      ball.updateAnimation(_animationController.value);
                      isClickAfter=false;
                      if ((ball.yVel>0)&&(ball.yVel.abs()* _animationController.value*baseTime +
ball.yPos + ball.ballRad >= 300)) {
                        ball.mulYvel(-0.7);
                        print("${ball.yVel}, ${ball.yPos}");
                        ball.outVel();

                      }
                    }
                  }
                  return Container(
                    width: 300,
                    height: 300,
                    color: Colors.white70,
                    child: CustomPaint(
                      painter: _paint(ballPath: ball.draw),
                    ),
                  );
                }
            ),
          )
        )
    );
  }
}

class _paint extends CustomPainter {
  final Path ballPath;
```

```
  _paint({
    required this.ballPath,
  });

  @override
  void paint(Canvas canvas, Size size) {
    Paint paint = Paint()
      ..color = Colors.brown
      ..style = PaintingStyle.stroke
      ..strokeWidth = 2;

    Path path = Path();
    path.addPath(ballPath, Offset.zero);

    canvas.drawPath(path, paint);
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) => true;

}


class myBall{
  late double xPos;
  late double yPos;
  late double xVel;
  late double yVel;
  late double ballRad;
  late Path draw;
  double baseTime = 0.002;

  myBall.origin(){
    xPos=100;
    yPos=100;
    xVel=0;
    yVel=0;
    ballRad=20;
    draw=Path();
    for(double i=0; i<ballRad-1; i++){
      draw.addOval(Rect.fromCircle(
          center: Offset(
              xPos, yPos
          ),
          radius: i
      ));
```

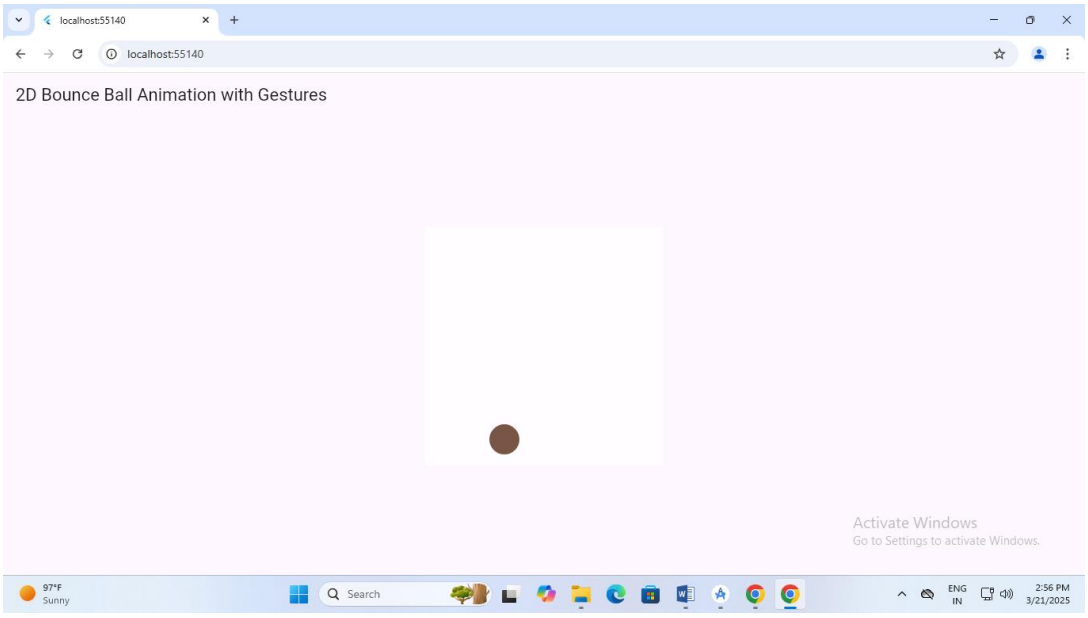
```
  }
}

void addXpos(double x){
  xPos+=x;
}

void subXpos(double x){
  xPos-=x;
}

void addYpos(double y){
  yPos+=y;
}

void subYpos(double y){
  yPos-=y;
}

void addXvel(double x){
  xVel+=x;
}

void subXvel(double x){
  xVel-=x;
}

void addYvel(double y){
  yVel+=y;
}

void subYvel(double y){
  yVel-=y;
}


void mulXvel(double v){
  xVel*=v;
}

void mulYvel(double v){
  yVel*=v;
}

void stop(){
  xVel=0;
```

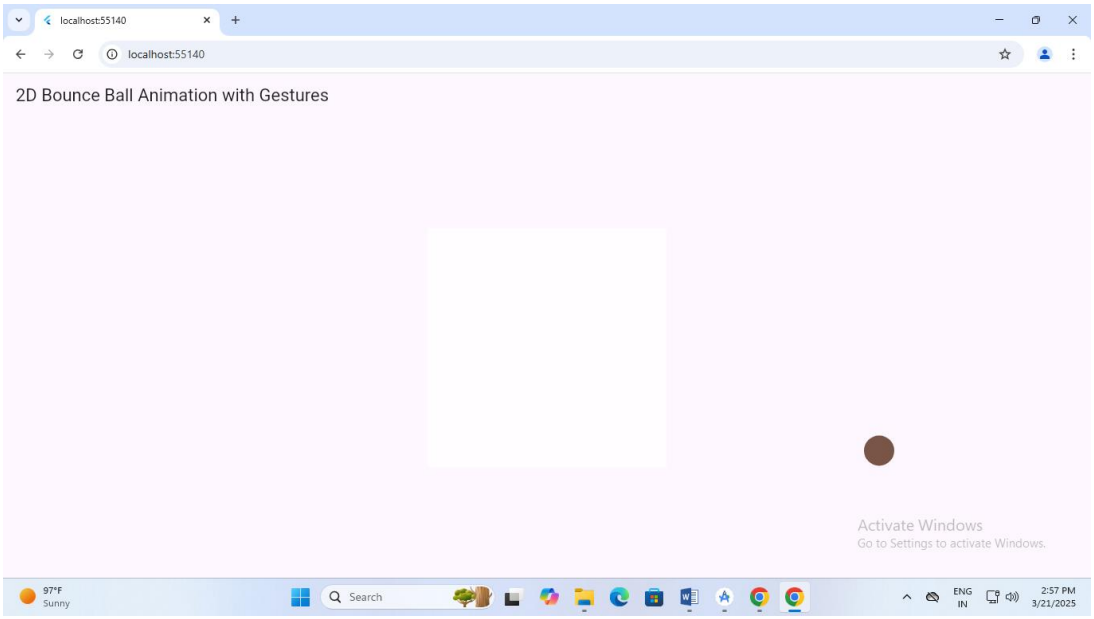
```
  yVel=0;
}

void outVel(){
  if(yVel.abs()<10){
    yVel=0;
  }
  if(xVel.abs()<10){
    xVel=0;
  }
}

void setPosition(double x, double y){
  xPos=x;
  yPos=y;
}

bool isBallRegion(double checkX, double checkY){
  if((pow(xPos-checkX, 2)+pow(yPos-checkY, 2))<=pow(ballRad, 2)){
    return true;
  }
  return false;
}

void updateDraw(){
  draw=Path();
  for(double i=0; i<ballRad-1; i++){
    draw.addOval(Rect.fromCircle(
        center: Offset(
          xPos,
          yPos,
        ),
        radius: i
    ));
  }
}

void updateAnimation(double animationValue){
  draw=Path();
  for(double i=0; i<ballRad-1; i++){
    draw.addOval(Rect.fromCircle(
        center: Offset(
          xPos + animationValue*xVel*baseTime,
          yPos + animationValue*yVel*baseTime,
        ),
        radius: i
```

```
    ));
   }
  }
}
```

8. **Save the file game2d.dart (click main menu -> saveall)**
9. **Select device as 'chrome(web)'**
10. **Click on run/debug configuration -> edit configurations ->specify dart file name (game2d.dart) -> browse and set dart entrypoint as game2d.dart -> click ok -> click run**

Output


localhost:55140
localhost:55140
2D Bounce Ball Animation with Gestures
Activate Windows
Go to Settings to activate Windows.
97°F
Sunny
Search
ENG
IN
2:56 PM
3/21/2025


localhost:55140
localhost:55140
2D Bounce Ball Animation with Gestures
Activate Windows
Go to Settings to activate Windows.
97°F
Sunny
Search
ENG
IN
2:57 PM
3/21/2025

Result

Thus, a gaming application with 2d-animations and gestures in flutter has been developed.