```
import 'package:flame/
game.dart';
import 'package:flame/
components.dart';
import 'package:flame/
input.dart';
import 'package:flame/
extensions.dart';
import 'package:flutter/
material.dart';
import 'dart:math';

void main() {
  runApp(GameWidget(game:
ShootingGame()));
```

```
}

class ShootingGame extends
FlameGame with
HasCollidables,
HasTappables,
HasDraggables,
HasKeyboardHandlerCompon
ents {
  late Player player;
  late Timer enemySpawner;
  int score = 0;
  int playerHealth = 3;
  TextComponent scoreText =
TextComponent(text: 'Score:
```

```
0');
  TextComponent healthText =
TextComponent(text: 'Health:
3');

  @override
  Future<void> onLoad() async
{
    player = Player();
    add(player);

    scoreText.position =
Vector2(10, 10);
    add(scoreText);
```

```
    healthText.position =
Vector2(10, 40);
    add(healthText);

    enemySpawner = Timer(1,
repeat: true, onTick: () {
      add(Enemy());
    });
    enemySpawner.start();
  }

  @override
  void update(double dt) {
    super.update(dt);
    enemySpawner.update(dt);
```

```
  }

  void increaseScore() {
    score += 10;
    scoreText.text = 'Score:
$score';
  }

  void decreaseHealth() {
    playerHealth -= 1;
    healthText.text = 'Health:
$playerHealth';
    if (playerHealth <= 0) {
      gameOver();
    }
```

```
  }

  void gameOver() {
    pauseEngine();
    add(GameOverText());
  }
}

class Player extends
SpriteComponent with
HasGameRef<ShootingGame>
, KeyboardHandler {
  Player() : super(size:
Vector2(50, 50));
```

```
  @override
  Future<void> onLoad() async
{
    position = gameRef.size / 2;
    sprite = await
gameRef.loadSprite('player.pn
g');
  }

  @override
  void update(double dt) {
    super.update(dt);
  }

  @override
```

```
  bool
onKeyEvent(RawKeyEvent
event,
Set<LogicalKeyboardKey>
keysPressed) {
    if (event is
RawKeyDownEvent) {
      if (event.logicalKey ==
LogicalKeyboardKey.arrowLeft
) {
        move(Vector2(-10, 0));
      } else if (event.logicalKey
==
LogicalKeyboardKey.arrowRig
ht) {
```

```
        move(Vector2(10, 0));
      } else if (event.logicalKey
==
LogicalKeyboardKey.space) {
        shoot();
      }
    }
    return true;
  }

  void move(Vector2 delta) {
    position.add(delta);

position.clamp(Vector2.zero()
+ size / 2, gameRef.size - size
```

```
/ 2);
  }

  void shoot() {
    Bullet bullet =
Bullet(position:
position.clone());
    gameRef.add(bullet);
  }
}

class Enemy extends
SpriteComponent with
HasGameRef<ShootingGame>
 {
```

```
  double speed = 100;
  Enemy() : super(size:
Vector2(40, 40));

  @override
  Future<void> onLoad() async
{
    position =
Vector2(Random().nextDouble
() * gameRef.size.x, 0);
    sprite = await
gameRef.loadSprite('enemy.pn
g');
  }
```

```
  @override
  void update(double dt) {
    super.update(dt);
    position.y += speed * dt;

    if (position.y >
gameRef.size.y) {
      removeFromParent();
      gameRef.decreaseHealth();
    }
  }
}

class Bullet extends
RectangleComponent with
```

```
HasGameRef<ShootingGame>
 {
  Bullet({required Vector2
position}) : super(position:
position, size: Vector2(5, 20),
paint: Paint()..color =
Colors.yellow);

  @override
  void update(double dt) {
    super.update(dt);
    position.y -= 300 * dt;

    if (position.y < 0) {
      removeFromParent();
```

```
    }

    for (var enemy in
gameRef.children.whereType<
Enemy>()) {
      if
(enemy.toRect().overlaps(toRe
ct())) {

enemy.removeFromParent();
        removeFromParent();
        gameRef.increaseScore();
        break;
      }
    }
```

```
  }
}

class GameOverText extends
TextComponent with
HasGameRef<ShootingGame>
 {
  GameOverText() : super(text:
'Game Over', textRenderer:
TextPaint(style:
TextStyle(fontSize: 48, color:
Colors.red)));

  @override
  Future<void> onLoad() async
```

```
{
    position = gameRef.size / 2
- Vector2(100, 24);
  }
}
```