Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Рубежный контроль № 1

по курсу «Разработка мобильных приложений»

«Работа с библиотекой flutter cube»

Студент группы ИУ9-72Б Шемякин В.А.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

Реализовать мобильное приложение, выводящее трехмерный объект по вариантам. Использование библиотеки на усмотрение программиста из рассмотренных на лекции.

Вариант: Реализовать приложение движения камеры по трем осям внутри пространства наполненного кубами разного радиуса разбросанных хаотически, количество сфер задается из формы.

2 Результаты

Исходный код программы представлен в листинге 1.

```
import 'dart:math' as math;
import 'package:flutter/material.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter_cube/flutter_cube.dart';

void main() => runApp(const App());

class App extends StatelessWidget {
  const App({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: '                          ',
      theme: ThemeData.dark(),
      home: const HomePage(title: 'Cubes and camera'),
    );
  }
}

class HomePage extends StatefulWidget {
  const HomePage({super.key, this.title});
  final String? title;

  @override
  State<HomePage> createState() => HomePageState();
}
```

```
29
30 class HomePageState extends State<HomePage> with
       SingleTickerProviderStateMixin {
31   Scene? scene;
32   Object? root;
33
34   final List<Collider> colliders = <Collider>[];
35   final double camRadius = 1.2;
36
37   final TextEditingController countCtrl = TextEditingController(
       text: '150');
38   final TextEditingController worldCtrl = TextEditingController(
       text: '60');
39   int count = 150;
40   double worldHalf = 60;
41   int seed = 42;
42
43   double yaw = 0;
44   double pitch = 0;
45   double fov = 60;
46
47   late Vector3 camPos;
48   Vector3 camTarget = Vector3(0, 0, 0);
49
50   late AnimationController controller;
51
52   Vector3? boundsMin;
53   Vector3? boundsMax;
54   double extraOut = 15.0;
55
56   @override
57   void initState() {
58     super.initState();
59     camPos = Vector3(0, 0, 120);
60     controller = AnimationController(duration: const Duration(
       seconds: 20), vsync: this)
61       ..addListener(() {
62         final s = scene;
63         if (s != null) s.update();
64       })
65       ..repeat();
66   }
67
68   @override
69   void dispose() {
```

```
70      controller.dispose();
71      super.dispose();
72    }
73
74    void onSceneCreated(Scene s) {
75      scene = s;
76      s.camera.fov = fov;
77      updateCamera();
78      buildWorld();
79    }
80
81    void buildWorld() {
82      final s = scene;
83      if (s == null) return;
84
85      if (root != null) {
86        s.world.remove(root!);
87        root = null;
88      }
89      colliders.clear();
90
91      final rng = math.Random(seed);
92      final r = Object(name: 'root');
93
94      for (int i = 0; i < count; i++) {
95        final x = (rng.nextDouble() * 2 - 1) * worldHalf;
96        final y = (rng.nextDouble() * 2 - 1) * worldHalf;
97        final z = (rng.nextDouble() * 2 - 1) * worldHalf;
98
99        final size = 0.5 + rng.nextDouble() * 5.0;
100       final pos = Vector3(x, y, z);
101       const variantPath = "assets/cube/cube.obj";
102       final cube = Object(
103         position: pos.clone(),
104         scale: Vector3.all(size),
105         backfaceCulling: false,
106         lighting: true,
107         fileName: variantPath,
108       );
109
110       cube.rotation.setValues(
111         rng.nextDouble() * 360,
112         rng.nextDouble() * 360,
113         rng.nextDouble() * 360,
114       );
```

```
115
116         final radius = size * math.sqrt(3) * 0.5;
117         colliders.add(Collider(center: pos.clone(), radius: radius));
118
119         r.add(cube);
120       }
121
122       root = r;
123       s.world.add(root!);
124       s.update();
125
126       computeBounds();
127     }
128
129     void computeBounds() {
130       if (colliders.isEmpty) {
131         boundsMin = Vector3(-worldHalf, -worldHalf, -worldHalf);
132         boundsMax = Vector3(worldHalf, worldHalf, worldHalf);
133         return;
134       }
135       double minX = double.infinity, minY = double.infinity, minZ =
        double.infinity;
136       double maxX = -double.infinity, maxY = -double.infinity, maxZ =
        -double.infinity;
137       for (final c in colliders) {
138         minX = math.min(minX, c.center.x - c.radius);
139         minY = math.min(minY, c.center.y - c.radius);
140         minZ = math.min(minZ, c.center.z - c.radius);
141         maxX = math.max(maxX, c.center.x + c.radius);
142         maxY = math.max(maxY, c.center.y + c.radius);
143         maxZ = math.max(maxZ, c.center.z + c.radius);
144       }
145       boundsMin = Vector3(minX, minY, minZ);
146       boundsMax = Vector3(maxX, maxY, maxZ);
147     }
148
149     void updateCamera() {
150       final s = scene;
151       if (s == null) return;
152       s.camera.target.setFrom(camTarget);
153       s.camera.position.setFrom(camPos);
154       s.camera.fov = fov;
155     }
156
157     Vector3 forward() {
```

```
158     final cy = math.cos(yaw), sy = math.sin(yaw);
159     final cp = math.cos(pitch), sp = math.sin(pitch);
160     return Vector3(-sy * cp, sp, -cy * cp)..normalize();
161   }
162
163   Vector3 right() {
164     final f = forward();
165     final up = Vector3(0, 1, 0);
166     final r = up.cross(f)..normalize();
167     return r;
168   }
169
170   Vector3 up() {
171     final r = right();
172     final f = forward();
173     final u = f.cross(r)..normalize();
174     return u;
175   }
176
177   void moveLocal({double dx = 0, double dy = 0, double dz = 0}) {
178     final r = right();
179     final u = up();
180     final f = forward();
181     final delta = r * dx + u * dy + f * dz;
182
183     final proposed = camPos + delta;
184     final resolved = resolveCollisions(proposed);
185
186     camTarget += (resolved - camPos);
187     camPos = resolved;
188
189     updateCamera();
190     setState(() {});
191   }
192
193   void rotateCamera(double dYaw, double dPitch) {
194     const double limit = math.pi / 2 - 0.01;
195     yaw += dYaw;
196     pitch = (pitch + dPitch).clamp(-limit, limit);
197     updateCamera();
198     setState(() {});
199   }
200
201   Offset? lastDrag;
202   double lastScale = 1.0;
```

```
203
204   void onScaleStart(ScaleStartDetails d) {
205     lastDrag = d.focalPoint;
206     lastScale = 1.0;
207   }
208
209   void onScaleUpdate(ScaleUpdateDetails d) {
210     final pos = d.focalPoint;
211     if (lastDrag != null) {
212       final delta = pos - lastDrag!;
213       rotateCamera(delta.dx * 0.005, delta.dy * 0.005);
214     }
215     lastDrag = pos;
216
217     final scaleDelta = d.scale - lastScale;
218     if (scaleDelta.abs() > 0.01) {
219       moveLocal(dz: scaleDelta * 25);
220     }
221     lastScale = d.scale;
222   }
223
224   void onScaleEnd(ScaleEndDetails d) {
225     lastDrag = null;
226     lastScale = 1.0;
227   }
228
229   void regenerate() {
230     final parsedCount = int.tryParse(countCtrl.text.trim());
231     final parsedWorld = double.tryParse(worldCtrl.text.trim());
232     count = (parsedCount == null || parsedCount < 0) ? 0 :
      parsedCount.clamp(0, 5000);
233     worldHalf = (parsedWorld == null || parsedWorld <= 0) ? 60 :
      parsedWorld;
234     seed = DateTime.now().millisecondsSinceEpoch & 0xFFFF;
235     buildWorld();
236     setState(() {});
237   }
238
239   @override
240   Widget build(BuildContext context) {
241     return Scaffold(
242       appBar: AppBar(title: Text(widget.title ?? '3D Cubes')),
243       body: Column(
244         children: [
245           Padding(
```

```
246             padding: const EdgeInsets.all(8.0),
247             child: Wrap(
248               crossAxisAlignment: WrapCrossAlignment.center,
249               spacing: 8,
250               runSpacing: 8,
251               children: [
252                 const Text('                                    '),
253                 SizedBox(
254                   width: 90,
255                   child: TextField(
256                     controller: countCtrl,
257                     keyboardType: TextInputType.number,
258                     decoration: const InputDecoration(isDense: true
    , hintText: '        . 200'),
259                     onSubmitted: (_) => regenerate(),
260                   ),
261                 ),
262                 const Text('              '),
263                 SizedBox(
264                   width: 90,
265                   child: TextField(
266                     controller: worldCtrl,
267                     keyboardType: const TextInputType.
    numberWithOptions(decimal: true),
268                     decoration: const InputDecoration(isDense: true
    , hintText: '        . 60'),
269                     onSubmitted: (_) => regenerate(),
270                   ),
271                 ),
272                 FilledButton.icon(
273                   onPressed: regenerate,
274                   icon: const Icon(Icons.casino),
275                   label: const Text('                              '),
276                 ),
277                 const SizedBox(width: 16),
278                 moveBtn('X-', () => moveLocal(dx: -5)),
279                 moveBtn('X+', () => moveLocal(dx: 5)),
280                 moveBtn('Y+', () => moveLocal(dy: 5)),
281                 moveBtn('Y-', () => moveLocal(dy: -5)),
282                 moveBtn('Z-', () => moveLocal(dz: -5)),
283                 moveBtn('Z+', () => moveLocal(dz: 5)),
284               ],
285             ),
286           ),
287           Expanded(
```

```
288             child: Listener(
289               onPointerSignal: (signal) {
290                 if (signal is PointerScrollEvent) {
291                   moveLocal(dz: -signal.scrollDelta.dy * 0.5);
292                 }
293               },
294               child: GestureDetector(
295                 onScaleStart: onScaleStart,
296                 onScaleUpdate: onScaleUpdate,
297                 onScaleEnd: onScaleEnd,
298                 child: Cube(
299                   onSceneCreated: onSceneCreated,
300                 ),
301               ),
302             ),
303           ),
304           Padding(
305             padding: const EdgeInsets.symmetric(horizontal: 12,
    vertical: 8),
306             child: Row(
307               children: [
308                 Text('Cam: (${camPos.x.toStringAsFixed(1)}, ${
    camPos.y.toStringAsFixed(1)}, ${camPos.z.toStringAsFixed(1)})')
    ,
309                 const Spacer(),
310               ],
311             ),
312           ),
313         ],
314       ),
315     );
316   }
317
318   Vector3 resolveCollisions(Vector3 proposed) {
319     var corrected = proposed.clone();
320     for (final c in colliders) {
321       final toCenter = corrected - c.center;
322       final dist = toCenter.length;
323       final minDist = camRadius + c.radius + 0.1;
324       if (dist < minDist) {
325         if (dist == 0) {
326           corrected.x = c.center.x + minDist;
327         } else {
328           final push = (minDist - dist);
329           final n = toCenter / dist;
```

```
330             corrected += n * push;
331           }
332         }
333       }
334
335       final minB = boundsMin;
336       final maxB = boundsMax;
337       if (minB != null && maxB != null) {
338         corrected.x = corrected.x.clamp(minB.x - extraOut, maxB.x +
      extraOut);
339         corrected.y = corrected.y.clamp(minB.y - extraOut, maxB.y +
      extraOut);
340         corrected.z = corrected.z.clamp(minB.z - extraOut, maxB.z +
      extraOut);
341       } else {
342         corrected.x = corrected.x.clamp(-worldHalf * 1.5, worldHalf *
       1.5);
343         corrected.y = corrected.y.clamp(-worldHalf * 1.5, worldHalf *
       1.5);
344         corrected.z = corrected.z.clamp(-worldHalf * 1.5, worldHalf *
       1.5);
345       }
346       return corrected;
347     }
348
349     Widget moveBtn(String label, VoidCallback onTap) {
350       return OutlinedButton(onPressed: onTap, child: Text(label));
351     }
352 }
353
354 class Collider {
355   Collider({required this.center, required this.radius});
356   final Vector3 center;
357   final double radius;
358 }
```

Результат запуска представлен на рисунке 1.

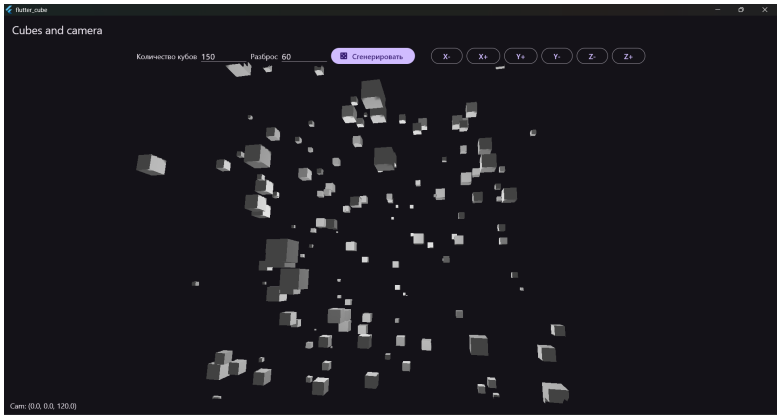

Рис. 1 — Результат

.