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»

Лабораторная работа № 7

по курсу «Разработка мобильных приложений»

«Визуализация 3D объекта руки с возможностью захвата объекта»

Студент группы ИУ9-72Б Шемякин В.А.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

1. Реализовать 3D-модель руки согласно видео.
2. Добавить в сцену любой OBJ объект — это может быть сфера, куб, череп или иная фигура.
3. Реализовать хватание кистью руки объекта из п.2 и перемещение по сцене.

2 Результаты

Исходный код программы представлен в листинге 1.

```
1  import 'package:ditredi/ditredi.dart';
2  import 'package:flutter/material.dart';
3  import 'package:vector_math/vector_math_64.dart' as vector;
4  import 'dart:math';
5
6  void main() {
7    runApp(const MyApp());
8  }
9
10 class MyApp extends StatefulWidget {
11   const MyApp({Key? key}) : super(key: key);
12
13   @override
14   State<MyApp> createState() => _MyAppState();
15 }
16
17 class _MyAppState extends State<MyApp> {
18   var indexAngle = 0.0;
19   var middleAngle = 0.0;
20   var ringAngle = 0.0;
21   var pinkyAngle = 0.0;
22
23   var handX = 0.0;
24   var handY = 0.0;
25   var handZ = 0.0;
26
27   var earthX = 0.0;
28   var earthY = 40.0;
29   var earthZ = 0.0;
30
```

```
31  final double grabRadius = 15.0;
32  final double collisionRadius = 5.0;
33  bool isGrabbed = false;
34  double grabbedOffsetX = 0.0;
35  double grabbedOffsetY = 0.0;
36  double grabbedOffsetZ = 0.0;
37
38  final Future<List<Mesh3D>> sphere = _generatePoints();
39
40  bool canGrab() {
41    final distance = sqrt(
42      pow(handX - earthX, 2) +
43      pow(handY - earthY, 2) +
44      pow(handZ - earthZ, 2)
45    );
46    return distance <= grabRadius;
47  }
48
49  bool checkCollision(double newHandX, double newHandY, double
     newHandZ) {
50    final distance = sqrt(
51      pow(newHandX - earthX, 2) +
52      pow(newHandY - earthY, 2) +
53      pow(newHandZ - earthZ, 2)
54    );
55    if (!isGrabbed) {
56      return distance < collisionRadius;
57    }
58    return false;
59  }
60
61  final _controller = DiTreDiController(
62    rotationX: 0,
63    rotationY: 0,
64    light: vector.Vector3(-0.5, -0.5, 0.5),
65    maxUserScale: 10,
66    minUserScale: 0.5,
67    userScale: 1,
68  );
69
70  @override
71  Widget build(BuildContext context) {
72    return MaterialApp(
73      darkTheme: ThemeData.dark(),
74      title: 'DiTreDi Demo',
```

```
75        theme: ThemeData(
76          primarySwatch: Colors.blue,
77        ),
78        home: Scaffold(
79          body: SafeArea(
80            child: Flex(
81              crossAxisAlignment: CrossAxisAlignment.start,
82              direction: Axis.vertical,
83              children: [
84                FutureBuilder(
85                    future: sphere,
86                    builder: (BuildContext context, AsyncSnapshot<
    List<Mesh3D>> snapshot){
87                      List<Widget> children;
88                      if(snapshot.hasData) {
89                        children = <Widget>[
90                          Expanded(
91                            child: DiTreDiDraggable(
92                              controller: _controller,
93                              child: DiTreDi(
94                                figures: [
95                                  TransformModifier3D(
96                                      snapshot.data![0],
97                                      Matrix4.identity()
98                                        ..translate(handX, handY,
    handZ)
99                                        ..rotateX(-pi/2)
100                                 ),
101                                 TransformModifier3D(
102                                     snapshot.data![1],
103                                     Matrix4.identity()
104                                       ..translate(handX, handY,
    handZ)
105                                       ..rotateX(-pi/2)
106                                       ..translate(3.05,1.15,8.75)
107                                       ..translate(-0.2,-0.25, -2.2)
108                                       ..rotateX(-(indexAngle * pi
    /18))
109                                       ..translate(0.2,0.25, 2.2)
110                                 ),
111                                 TransformModifier3D(
112                                     snapshot.data![2],
113                                     Matrix4.identity()
114                                       ..translate(handX, handY,
    handZ)
```

```
115                                         ..rotateX(-pi/2)
116                                         ..translate(0.7,0.0,9.75)
117                                         ..translate(0.0,-0.5, -2.25)
118                                         ..rotateX(-(middleAngle * pi
    /18))
119                                         ..translate(0.0,0.5, 2.25)
120                                     ),
121                                     TransformModifier3D(
122                                         snapshot.data![3],
123                                         Matrix4.identity()
124                                           ..translate(handX, handY,
    handZ)
125                                           ..rotateX(-pi/2)
126                                           ..translate(-2.0,-0.56,9.1)
127                                           ..translate(0.0,-0.25, -2.2)
128                                           ..rotateX(-(ringAngle * pi
    /18))
129                                           ..translate(0.0, 0.25, 2.2)
130                                           ..rotate
131                                     ),
132                                     TransformModifier3D(
133                                         snapshot.data![4],
134                                         Matrix4.identity()
135                                           ..translate(handX, handY,
    handZ)
136                                           ..rotateX(-pi/2)
137                                           ..translate(-4.65,-1.0,7.15)
138                                           ..translate(0.0,0.0, -1.25)
139                                           ..rotateX(-(pinkyAngle * pi
    /18))
140                                           ..translate(0.0,0.0, 1.25)
141                                     ),
142                                     TransformModifier3D(
143                                         snapshot.data![5],
144                                         Matrix4.identity()
145                                           ..translate(earthX, earthY,
    earthZ)
146                                           ..rotateX(-pi/2)
147                                           ..rotateY(pi)
148                                     ),
149                                   ],
150                                   controller: _controller,
151                                 ),
152                               ),
153                             ),
```

```
154                         const Padding(
155                           padding: EdgeInsets.all(8.0),
156                           child: Text("Drag to rotate. Scroll to
    zoom"),
157                         ),
158                         Expanded(
159   child: Column(
160     mainAxisSize: MainAxisSize.min,
161     children: [
162       //

163       const SizedBox(height: 10),
164       const Text("                                                ",
    style: TextStyle(fontWeight: FontWeight.bold)),
165       const SizedBox(height: 10),
166       Row(
167         mainAxisAlignment: MainAxisAlignment.center,
168         children: [
169           Column(
170             children: [
171               ElevatedButton(
172                 onPressed: () {
173                   setState(() {
174                     final newY = handY + 4.0;
175                     if (!checkCollision(handX, newY, handZ)) {
176                       handY = newY;
177                       if (isGrabbed) earthY = handY +
    grabbedOffsetY;
178                     }
179                   });
180                 },
181                 child: const Icon(Icons.arrow_upward),
182               ),
183               Row(
184                 mainAxisAlignment: MainAxisAlignment.center,
185                 children: [
186                   ElevatedButton(
187                     onPressed: () {
188                       setState(() {
189                         final newX = handX - 4.0;
190                         if (!checkCollision(newX, handY, handZ)) {
191                           handX = newX;
192                           if (isGrabbed) earthX = handX +
    grabbedOffsetX;
193                         }
```

```
194                       });
195                     },
196                     child: const Icon(Icons.arrow_back),
197                   ),
198                   const SizedBox(width: 10),
199                   ElevatedButton(
200                     onPressed: () {
201                       setState(() {
202                         final newX = handX + 4.0;
203                         if (!checkCollision(newX, handY, handZ)) {
204                           handX = newX;
205                           if (isGrabbed) earthX = handX +
    grabbedOffsetX;
206                         }
207                       });
208                     },
209                     child: const Icon(Icons.arrow_forward),
210                   ),
211                 ],
212               ),
213               ElevatedButton(
214                 onPressed: () {
215                   setState(() {
216                     final newY = handY - 4.0;
217                     if (!checkCollision(handX, newY, handZ)) {
218                       handY = newY;
219                       if (isGrabbed) earthY = handY +
    grabbedOffsetY;
220                     }
221                   });
222                 },
223                 child: const Icon(Icons.arrow_downward),
224               ),
225             ],
226           ),
227           const SizedBox(width: 30),
228           Column(
229             children: [
230               const Text("Z", style: TextStyle(fontSize: 12,
    fontWeight: FontWeight.bold)),
231               const SizedBox(height: 5),
232               ElevatedButton(
233                 onPressed: () {
234                   setState(() {
235                     final newZ = handZ + 4.0;
```

```
236                     if (!checkCollision(handX, handY, newZ)) {
237                       handZ = newZ;
238                       if (isGrabbed) earthZ = handZ +
    grabbedOffsetZ;
239                     }
240                   });
241                 },
242                 style: ElevatedButton.styleFrom(padding: const
    EdgeInsets.symmetric(horizontal: 20, vertical: 15)),
243                 child: const Text("   ", style: TextStyle(fontSize:
     16)),
244               ),
245               const SizedBox(height: 5),
246               ElevatedButton(
247                 onPressed: () {
248                   setState(() {
249                     final newZ = handZ - 4.0;
250                     if (!checkCollision(handX, handY, newZ)) {
251                       handZ = newZ;
252                       if (isGrabbed) earthZ = handZ +
    grabbedOffsetZ;
253                     }
254                   });
255                 },
256                 style: ElevatedButton.styleFrom(padding: const
    EdgeInsets.symmetric(horizontal: 20, vertical: 15)),
257                 child: const Text("   ", style: TextStyle(fontSize:
     16)),
258               ),
259             ],
260           ),
261         ],
262       ),
263
264       const SizedBox(height: 15),
265
266       //                    GRAB/RELEASE
267       const Text("                                        ", style:
    TextStyle(fontWeight: FontWeight.bold)),
268       const SizedBox(height: 10),
269       Row(
270         mainAxisAlignment: MainAxisAlignment.center,
271         children: [
272           ElevatedButton(
273             onPressed: () {
```

```
274               setState(() {
275                 indexAngle = middleAngle = ringAngle = pinkyAngle =
      12.0;
276                 if (canGrab() && !isGrabbed) {
277                   isGrabbed = true;
278                   grabbedOffsetX = earthX - handX;
279                   grabbedOffsetY = earthY - handY;
280                   grabbedOffsetZ = earthZ - handZ;
281                 }
282               });
283             },
284             style: ElevatedButton.styleFrom(
285               backgroundColor: Colors.red,
286               padding: const EdgeInsets.symmetric(horizontal: 30,
      vertical: 15),
287             ),
288             child: const Text("GRAB", style: TextStyle(fontSize:
      18, fontWeight: FontWeight.bold)),
289           ),
290           const SizedBox(width: 20),
291           ElevatedButton(
292             onPressed: () {
293               setState(() {
294                 indexAngle = middleAngle = ringAngle = pinkyAngle =
      0.0;
295                 isGrabbed = false;
296               });
297             },
298             style: ElevatedButton.styleFrom(
299               backgroundColor: Colors.green,
300               padding: const EdgeInsets.symmetric(horizontal: 30,
      vertical: 15),
301             ),
302             child: const Text("RELEASE", style: TextStyle(fontSize:
      18, fontWeight: FontWeight.bold)),
303           ),
304         ],
305       ),
306
307       const SizedBox(height: 15),
308
309       //
310       Slider(
311         value: indexAngle, min: 0, max: 12, divisions: 13,
312         label: (180 - 10 * indexAngle.round()).toString(),
```

```
313           onChanged: (v) => setState(() => indexAngle = v),
314         ),
315         Slider(
316           value: middleAngle, min: 0, max: 12, divisions: 13,
317           label: (180 - 10 * middleAngle.round()).toString(),
318           onChanged: (v) => setState(() => middleAngle = v),
319         ),
320         Slider(
321           value: ringAngle, min: 0, max: 12, divisions: 13,
322           label: (180 - 10 * ringAngle.round()).toString(),
323           onChanged: (v) => setState(() => ringAngle = v),
324         ),
325         Slider(
326           value: pinkyAngle, min: 0, max: 12, divisions: 13,
327           label: (180 - 10 * pinkyAngle.round()).toString(),
328           onChanged: (v) => setState(() => pinkyAngle = v),
329         ),
330       ],
331     ),
332 )
333                         ];
334                       }else{
335                         children = <Widget>[
336                           const Padding(
337                             padding: EdgeInsets.all(8.0),
338                             child: Text("Failed to load"),
339                           )
340                         ];
341                       }
342                       return Expanded(
343                         child: Column(
344                           mainAxisAlignment: MainAxisAlignment.center
    ,
345                           children: children,
346                         ),
347                       );
348                     }),
349               ],
350             ),
351           ),
352         ),
353       );
354     }
355 }
356
```

```
357 Future<List<Mesh3D>> _generatePoints() async{
358   return [
359     Mesh3D(await ObjParser().loadFromResources("assets/hand/hand.
      obj")),
360     Mesh3D(await ObjParser().loadFromResources("assets/hand/index.
      obj"),),
361     Mesh3D(await ObjParser().loadFromResources("assets/hand/middle.
      obj"),),
362     Mesh3D(await ObjParser().loadFromResources("assets/hand/ring.
      obj"),),
363     Mesh3D(await ObjParser().loadFromResources("assets/hand/pinky.
      obj"),),
364     Mesh3D(await ObjParser().loadFromResources("assets/hand/skull.
      obj"),)
365   ];
366 }
```

Результат запуска представлен на рисунке 1.

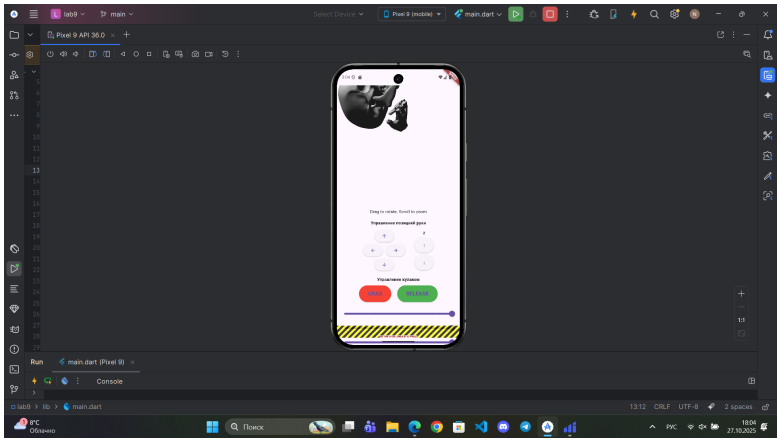

Рис. 1 — Результат

.