

# DEAKIN UNIVERSITY

## OBJECT ORIENTED DEVELOPMENT

### ONTRACK SUBMISSION

---

# A Simple Reaction-Timer Controller

---

*Submitted By:*

Peter STACEY

pstacey

2020/05/10 18:12

*Tutor:*

Dipto PRATYAKSA

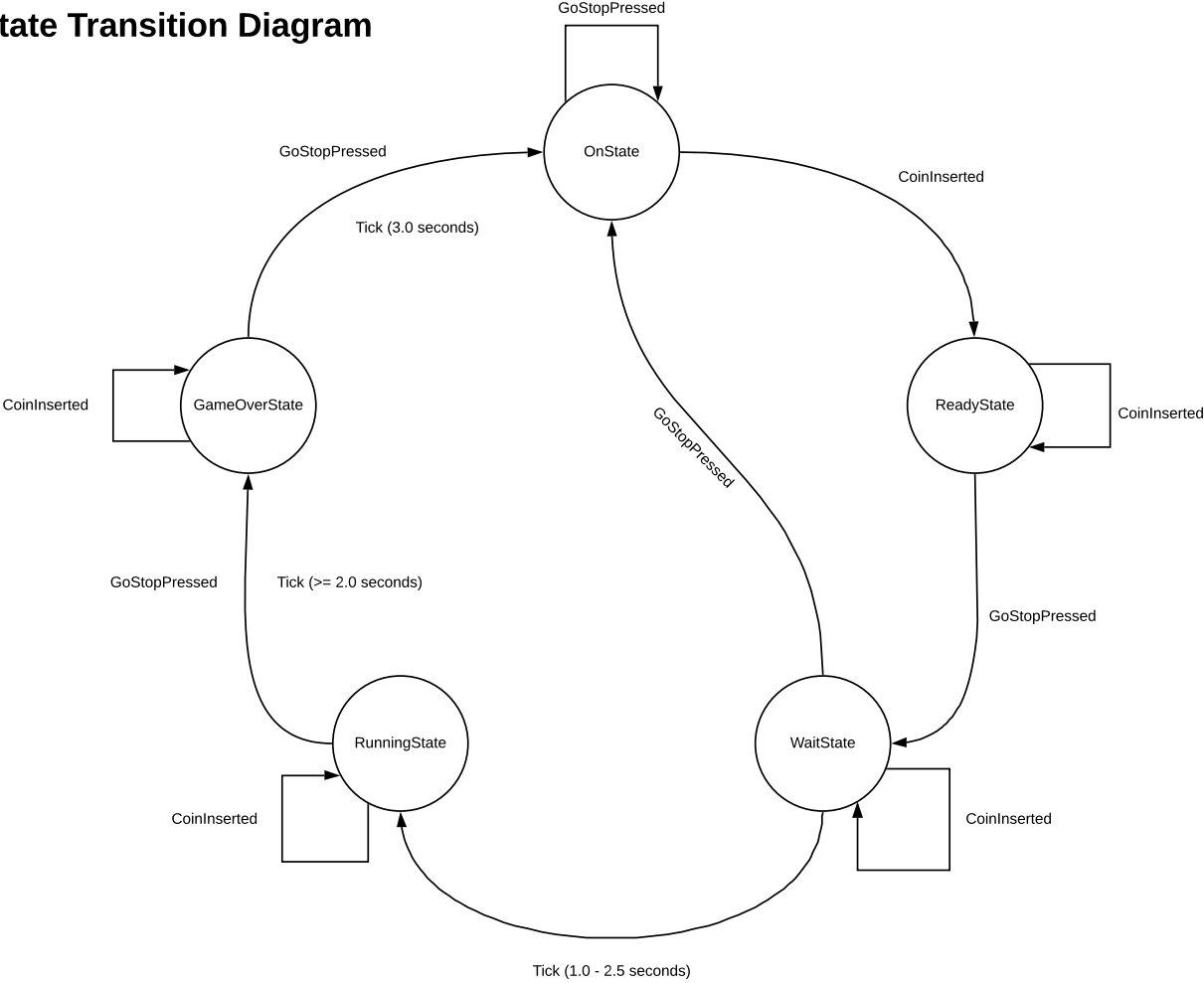
Outcome	Weight
Evaluate Code	◆◆◆◆◇
Principles	◆◆◆◇◇
Build Programs	◆◆◆◆◇
Design	◆◆◆◆◆
Justify	◆◆◆◇◇

By providing the base files, this task involved evaluating an existing program in order to understand how to integrate our additional file into it. It also involved applying the principle of OOP, to encapsulate data, abstract the design out to a recognised OOP design pattern, to implement the design through coding the required file using a State pattern and supporting our design with a diagram of the states and transitions. With the transition diagram and file, we have evidence of our effort and this is further supported by my video.

May 10, 2020



**SIT232 - Task 5.3D**  
**State Transition Diagram**



```
1  using SimpleReactionMachine;
2  using System;
3  using System.Data;
4
5  namespace SimpleReactionMachine
6  {
7      public class SimpleReactionController : IController
8      {
9          // Settings for the game times
10         private const int MIN_WAIT_TIME = 100; // Minimum wait time, 1 sec in ticks
11         private const int MAX_WAIT_TIME = 250; // Maximum wait time, 2.5 sec in
            → ticks
12         private const int MAX_GAME_TIME = 200; // Maximum of 2 sec to react, in
            → ticks
13         private const int GAMEOVER_TIME = 300; // Display result for 3 sec, in ticks
14         private const double TICKS_PER_SECOND = 100.0; // Based on 10ms ticks
15
16         // Instance variables and properties
17         private State _state;
18         private IGui Gui { get; set; }
19         private IRandom Rng { get; set; }
20         private int Ticks { get; set; }
21
22         /// <summary>
23         /// Connects the controller to the Gui and Random Number Generator
24         /// </summary>
25         /// <param name="gui">IGui concrete implementation</param>
26         /// <param name="rng">IRandom concrete implementation</param>
27         public void Connect(IGui gui, IRandom rng)
28         {
29             Gui = gui;
30             Rng = rng;
31             Init();
32         }
33
34         /// <summary>
35         /// Initialises the state of the controller at the start of the program
36         /// </summary>
37         public void Init()
38         {
39             _state = new OnState(this);
40         }
41
42         /// <summary>
43         /// Coin inserted event handler
44         /// </summary>
45         public void CoinInserted()
46         {
47             _state.CoinInserted();
48         }
49
50         /// <summary>
51         /// Go/Stop pressed event handler
```

```
52     /// </summary>
53     public void GoStopPressed()
54     {
55         _state.GoStopPressed();
56     }
57
58     /// <summary>
59     /// Tick event handler
60     /// </summary>
61     public void Tick()
62     {
63         _state.Tick();
64     }
65
66     /// <summary>
67     /// Sets the state of the controller to the desired state
68     /// </summary>
69     /// <param name="state">The new state to transition to</param>
70     private void SetState(State state)
71     {
72         _state = state;
73     }
74
75     /// <summary>
76     /// Base class for concrete State classes
77     /// </summary>
78     private abstract class State
79     {
80         protected SimpleReactionController _controller;
81
82         public State(SimpleReactionController controller)
83         {
84             _controller = controller;
85         }
86
87         public abstract void CoinInserted();
88         public abstract void GoStopPressed();
89         public abstract void Tick();
90     }
91
92     /// <summary>
93     /// State of the game when it is waiting for a coin to be inserted
94     /// </summary>
95     private class OnState : State
96     {
97         public OnState(SimpleReactionController controller) : base(controller)
98         {
99             _controller.Gui.SetDisplay("Insert coin");
100         }
101
102         public override void CoinInserted()
103         {
104             _controller.SetState(new ReadyState(_controller));
```

```
105         }
106         public override void GoStopPressed() { }
107         public override void Tick() { }
108     }
109
110     /// <summary>
111     /// State of the game when a coin has been inserted, but the game is not yet
112     /// started
113     /// </summary>
114     private class ReadyState : State
115     {
116         public ReadyState(SimpleReactionController controller) :
117             ↪ base(controller)
118         {
119             _controller.Gui.SetDisplay("Press Go!");
120         }
121
122         public override void CoinInserted() { }
123         public override void GoStopPressed()
124         {
125             _controller.SetState(new WaitState(_controller));
126         }
127         public override void Tick() { }
128     }
129
130     /// <summary>
131     /// State of the game when the game has started and it is waiting for the
132     /// random time
133     /// </summary>
134     private class WaitState : State
135     {
136         private int _waitTime;
137         public WaitState(SimpleReactionController controller) : base(controller)
138         {
139             _controller.Gui.SetDisplay("Wait...");
140             _controller.Ticks = 0;
141             _waitTime = _controller.Rng.GetRandom(MIN_WAIT_TIME, MAX_WAIT_TIME);
142         }
143
144         public override void CoinInserted() { }
145         public override void GoStopPressed()
146         {
147             _controller.SetState(new OnState(_controller));
148         }
149         public override void Tick()
150         {
151             _controller.Ticks++;
152             if(_controller.Ticks == _waitTime)
153             {
154                 _controller.SetState(new RunningState(_controller));
155             }
156         }
157     }
158 }
```

```
157
158     /// <summary>
159     /// State of the game when the timer is counting and it is waiting for the
160     /// user to react by pressing the Go/Stop button
161     /// </summary>
162     private class RunningState : State
163     {
164         public RunningState(SimpleReactionController controller) :
165             ↪ base(controller)
166         {
167             _controller.Gui.SetDisplay("0.00");
168             _controller.Ticks = 0;
169         }
170
171         public override void CoinInserted() { }
172         public override void GoStopPressed()
173         {
174             _controller.SetState(new GameOverState(_controller));
175         }
176
177         public override void Tick()
178         {
179             _controller.Ticks++;
180             _controller.Gui.SetDisplay(
181                 (_controller.Ticks / TICKS_PER_SECOND).ToString("0.00"));
182             if(_controller.Ticks == MAX_GAME_TIME)
183             {
184                 _controller.SetState(new GameOverState(_controller));
185             }
186         }
187     }
188
189     /// <summary>
190     /// State of the game when the time has expired, or the user reacted.
191     /// </summary>
192     private class GameOverState : State
193     {
194         public GameOverState(SimpleReactionController controller) :
195             ↪ base(controller)
196         {
197             _controller.Ticks = 0;
198         }
199
200         public override void CoinInserted() { }
201         public override void GoStopPressed()
202         {
203             _controller.SetState(new OnState(_controller));
204         }
205
206         public override void Tick()
207         {
208             _controller.Ticks++;
209             if(_controller.Ticks == GAMEOVER_TIME)
210             {
211                 _controller.SetState(new OnState(_controller));
212             }
213         }
214     }
215 }
```

```
208         _controller.SetState(new OnState(_controller));
209     }
210 }
211 }
212 }
213 }
```