A.3 Practice 06 - Launch

A.3.1 parent.launch.py

```python
from launch_ros.substitutions import FindPackageShare

from launch import LaunchDescription
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import PathJoinSubstitution, TextSubstitution


def generate_launch_description():
    colors = {
        'background_r': '200'
    }

    return LaunchDescription([
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource([
                PathJoinSubstitution([
                    FindPackageShare('launch_test'),
                    'launch',
                    'substitutions1.launch.py'
                ])
            ]),
            launch_arguments={
                'turtlesim_ns': 'turtlesim2',
                'use_provided_red': 'True',
                'new_background_r': TextSubstitution(text=str(colors['background_r']))
            }.items()
        )
    ])
```

A.3.2 substitutions1.launch.py

```python
from launch_ros.actions import Node

from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, ExecuteProcess, TimerAction
from launch.conditions import IfCondition
from launch.substitutions import LaunchConfiguration, PythonExpression


def generate_launch_description():
    turtlesim_ns = LaunchConfiguration('turtlesim_ns')
    use_provided_red = LaunchConfiguration('use_provided_red')
    new_background_r = LaunchConfiguration('new_background_r')

```

```
    turtlesim_ns_launch_arg = DeclareLaunchArgument(
        'turtlesim_ns',
        default_value='turtlesim1'
    )
    use_provided_red_launch_arg = DeclareLaunchArgument(
        'use_provided_red',
        default_value='False'
    )
    new_background_r_launch_arg = DeclareLaunchArgument(
        'new_background_r',
        default_value='200'
    )

    turtlesim_node = Node(
        package='turtlesim',
        namespace=turtlesim_ns,
        executable='turtlesim_node',
        name='sim'
    )
    spawn_turtle = ExecuteProcess(
        cmd=[[
            'ros2 service call ',
            turtlesim_ns,
            '/spawn ',
            'turtlesim/srv/Spawn ',
            '"{x: 2, y: 2, theta: 0.2}"'
        ]],
        shell=True
    )
    change_background_r = ExecuteProcess(
        cmd=[[
            'ros2 param set ',
            turtlesim_ns,
            '/sim background_r ',
            '120'
        ]],
        shell=True
    )
    change_background_r_conditioned = ExecuteProcess(
        condition=IfCondition(
            PythonExpression([
                new_background_r,
                ' == 200',
                ' and ',
                use_provided_red
            ])
        ),
        cmd=[[
            'ros2 param set ',
            turtlesim_ns,
            '/sim background_r ',
            new_background_r
        ]],
        shell=True
```

```
    )

    return LaunchDescription([
        turtlesim_ns_launch_arg,
        use_provided_red_launch_arg,
        new_background_r_launch_arg,
        turtlesim_node,
        spawn_turtle,
        change_background_r,
        TimerAction(
            period=2.0,
            actions=[change_background_r_conditioned],
        )
    ])
```

A.3.3 eventhandler.launch.py

```
from launch_ros.actions import Node

from launch import LaunchDescription
from launch.actions import (DeclareLaunchArgument, EmitEvent,
   ExecuteProcess,
                            LogInfo, RegisterEventHandler, TimerAction)
from launch.conditions import IfCondition
from launch.event_handlers import (OnExecutionComplete, OnProcessExit,
                                   OnProcessIO, OnProcessStart, OnShutdown)
from launch.events import Shutdown
from launch.substitutions import (EnvironmentVariable, FindExecutable,
                                  LaunchConfiguration, LocalSubstitution,
                                  PythonExpression)


def generate_launch_description():
    turtlesim_ns = LaunchConfiguration('turtlesim_ns')
    use_provided_red = LaunchConfiguration('use_provided_red')
    new_background_r = LaunchConfiguration('new_background_r')

    turtlesim_ns_launch_arg = DeclareLaunchArgument(
        'turtlesim_ns',
        default_value='turtlesim1'
    )
    use_provided_red_launch_arg = DeclareLaunchArgument(
        'use_provided_red',
        default_value='False'
    )
    new_background_r_launch_arg = DeclareLaunchArgument(
        'new_background_r',
        default_value='200'
    )

    turtlesim_node = Node(
        package='turtlesim',
        namespace=turtlesim_ns,
```

```
        executable='turtlesim_node',
        name='sim'
    )
    spawn_turtle = ExecuteProcess(
        cmd=[[
            FindExecutable(name='ros2'),
            ' service call ',
            turtlesim_ns,
            '/spawn ',
            'turtlesim/srv/Spawn ',
            '"{x: 2, y: 2, theta: 0.2}"'
        ]],
        shell=True
    )
    change_background_r = ExecuteProcess(
        cmd=[[
            FindExecutable(name='ros2'),
            ' param set ',
            turtlesim_ns,
            '/sim background_r ',
            '120'
        ]],
        shell=True
    )
    change_background_r_conditioned = ExecuteProcess(
        condition=IfCondition(
            PythonExpression([
                new_background_r,
                ' == 200',
                ' and ',
                use_provided_red
            ])
        ),
        cmd=[[
            FindExecutable(name='ros2'),
            ' param set ',
            turtlesim_ns,
            '/sim background_r ',
            new_background_r
        ]],
        shell=True
    )

    return LaunchDescription([
        turtlesim_ns_launch_arg,
        use_provided_red_launch_arg,
        new_background_r_launch_arg,
        turtlesim_node,
        RegisterEventHandler(
            OnProcessStart(
                target_action=turtlesim_node,
                on_start=[
                    LogInfo(msg='Turtlesim started, spawning turtle'),
                    spawn_turtle
```

```
                ]
            )
        ),
        RegisterEventHandler(
            OnProcessIO(
                target_action=spawn_turtle,
                on_stdout=lambda event: LogInfo(
                    msg='Spawn request says "{}"'.format(
                        event.text.decode().strip())
                )
            )
        ),
        RegisterEventHandler(
            OnExecutionComplete(
                target_action=spawn_turtle,
                on_completion=[
                    LogInfo(msg='Spawn finished'),
                    change_background_r,
                    TimerAction(
                        period=2.0,
                        actions=[change_background_r_conditioned],
                    )
                ]
            )
        ),
        RegisterEventHandler(
            OnProcessExit(
                target_action=turtlesim_node,
                on_exit=[
                    LogInfo(msg=(EnvironmentVariable(name='USER'),
                            ' closed the turtlesim window')),
                    EmitEvent(event=Shutdown(
                        reason='Window closed'))
                ]
            )
        ),
        RegisterEventHandler(
            OnShutdown(
                on_shutdown=[LogInfo(
                    msg=['Launch was asked to shutdown: ',
                        LocalSubstitution('event.reason')]
                )]
            )
        ),
    ])
```