VEHICLE SIMULATION MOTION RIDE INTERFACE PROTOCOL – DRAFT

Vehicle simulation motion ride control system will consist of two computers: Motion Computer (MC) and Show Computer (SC). Motion computer will be responsible for motion and overall control of the ride. MC will be master computer to start, stop and pause the game. MC will send these commands to SC through Ethernet packets.

SC will get user inputs (wheel, gas pedal, brakes, etc.) and simulate vehicle dynamics. SC will generate realistic visuals and audio. It will also send vehicle orientation data through Ethernet packet to MC at 30 Hz. MC will generate motion cues based on data from SC. This vehicle orientation data will consist of pitch and roll orientation of vehicle and linear velocities on vehicle coordinates. In vehicle coordinate system X coordinate represents vehicle's forward and backward movement, Y coordinate represents vehicle's left and right movement, and Z coordinate represents vehicles up and down movement. Typically speed up and speed downs generates speed on X, left and right turns generates speed on Y, and climb up and downs generates speed on Z.

Vehicle simulation motion ride will be powered up once in the morning and it will cycle throughout the day. These ride steps will correspond to ride states in SC computer as shown below:

- POWER_DOWN_STATE: SC is not ready yet.
- WAIT_FOR_DOOR_TO_OPEN_STATE: After end-of-game script played, door is still closed.
- LOAD _STATE: Guest will be loaded into ride. During this time load scene will run on SC.
- READY_TO_RUN_STATE: Operator will close ride door. SC will start playing ready to run scene.
- RUN_STATE: Operator will press run button and SC will play the simulation according to user inputs.
- STOP_END_OF_GAME: At the end of the game SC will play stop scene until ride doors open
- STOP_AUTO: If ride stops due to technical problems or operator or guest hits stop button, SC will play stop-auto scene.
- STOP_MANUAL: If operator presses home button while ride is running, SC will play stop-manual scene.

SC will transition states using commands coming from MC. These commands are described below as shown in Figure 2. State transitions for states are shown in Figure 3.

- START_LOAD_COMMAND: Sent when MC ready to start ride cycle.
- DOOR_CLOSED_COMMAND: Sent after ride door closed.
- DOOR_OPENED_COMMAND: Sent when ride door reopened before start of ride.
- GAME_START_COMMAND: Sent when operator presses run button.
- STOP_END_OF_GAME: Game will end at the end of predefined game duration. At the end of the duration MC will send this command.
- STOP_AUTO: System stops automatically due to technical problem in which operator needs to check the technical problem and restart the game if everything is OK.
- STOP_MANUAL: Operator stops the ride by pressing HOME button outside due to some reason.

```
enum STATES{
POWER_DOWN_STATE,
WAIT_FOR_DOR_TO_OPEN_STATE,
LOAD_STATE,
READY_TO_RUN_STATE,
RUN_STATE,
STOP_AUTO_STATE,
STOP_MANUAL_STATE,
STOP_END_OF_GAME_STATE,
NUMBER_OF_GAME_STATES
}
```

Figure 1: Ride States

```
enum COMMANDS {
NO_COMMAND,
START_LOAD_LANGUAGE_1_COMMAND,
DOOR_OPENED_COMMAND,
DOOR_CLOSED_COMMAND,
GAME_START_COMMAND,
STOP_END_OF_GAME_COMMAND,
STOP_AUTO_COMMAND,
STOP_MANUAL_COMMAND,
START_LOAD_LANGUAGE_2_COMMAND,
START_LOAD_LANGUAGE_3_COMMAND,
START_LOAD_DEMO_COMMAND,
NUMBER_OF_GAME_COMMAND};
```

Figure 2: Ride Commands

```
typedef struct MSG_FLIGHTSIM_DATA_STRUCT
{
    MSG_HEADER  msgHdr;
    Unsigned char ride_command;
     Unsigned char spare[3];
}, MSG_COMMAND;
```

Figure 3: MC to SC command message

```
typedef struct MSG_VEHICLE_MOTION_STRUCT
{
    MSG_HEADER  msg_header;
        double  x_lin_vel;
        double  y_lin_vel;
        double  z_lin_vel;
        double  pitch_pos;
        double  roll_pos;
} MSG_VEHICLE_MOTION_DATA;
```

Figure 4: SC to MC vehicle motion message

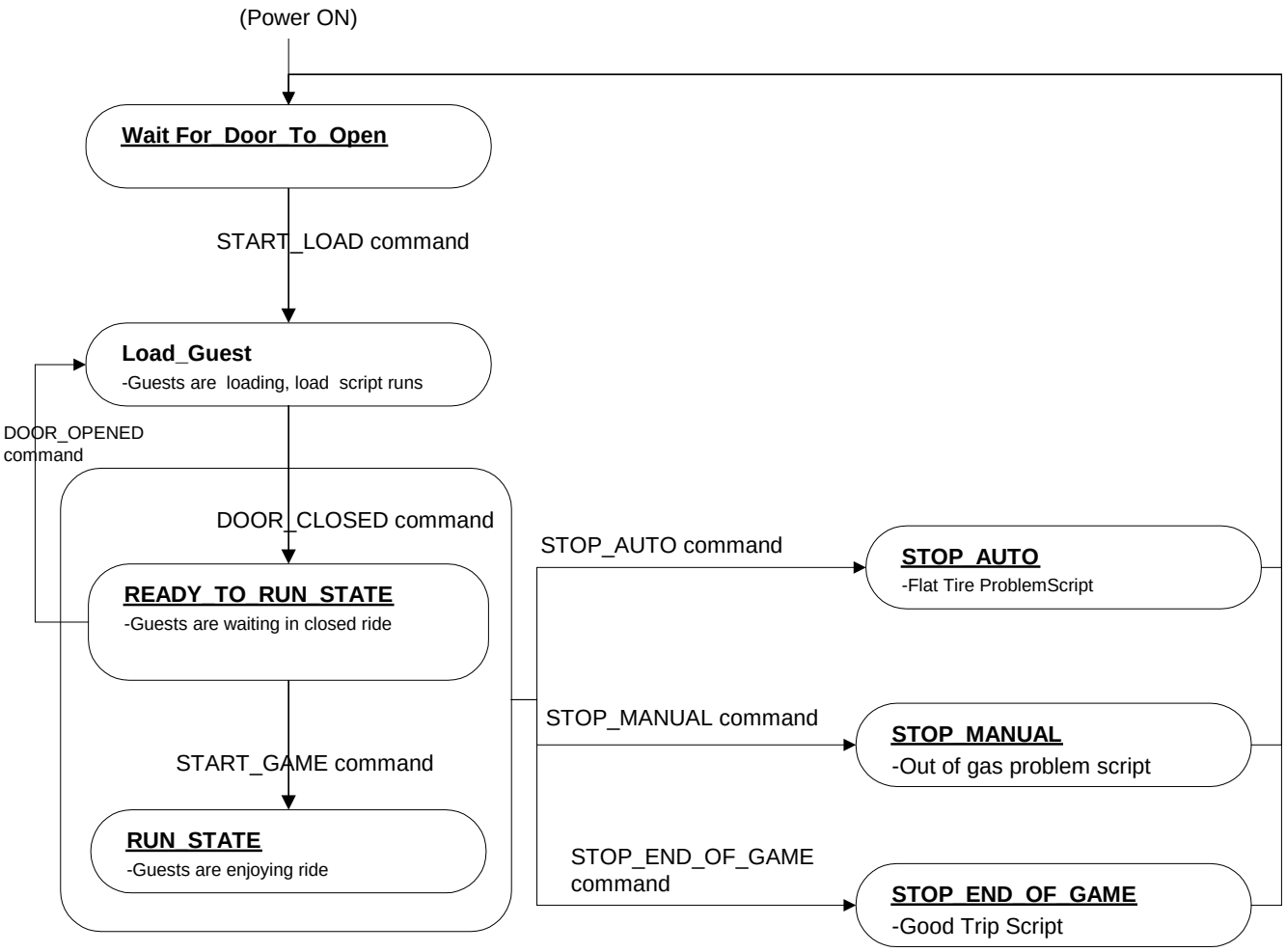

Figure 5: State Transitions