Part V: Distributed Programming

Marco T. Morazán

Seton Hall University

Outline

Intro to Distributed Programming

- You are probably very familiar with distributed programming as a user
- ext messaging systems, multiplayer video games, and social media apps

Intro to Distributed Programming

- You are probably very familiar with distributed programming as a user
- ext messaging systems, multiplayer video games, and social media apps
- You solicit services from another program (usually running on another computer)

Intro to Distributed Programming

- You are probably very familiar with distributed programming as a user
- ext messaging systems, multiplayer video games, and social media apps
- You solicit services from another program (usually running on another computer)
- Dividing a problem into several tasks and writing a program for each task that communicates with the programs for other tasks is called *distributed programming*
- The tasks cooperate to solve a problem
- Each task defines a *component* which is a program
- Each component itself may be divided into subtasks and may be solved using one or more computers

Intro to Distributed Programming

- You are probably very familiar with distributed programming as a user
- ext messaging systems, multiplayer video games, and social media apps
- You solicit services from another program (usually running on another computer)
- Dividing a problem into several tasks and writing a program for each task that communicates with the programs for other tasks is called *distributed programming*
- The tasks cooperate to solve a problem
- Each task defines a *component* which is a program
- Each component itself may be divided into subtasks and may be solved using one or more computers
- The components cooperate by communicating with each other using message-passing
- Messages are exchanged via a network (e.g., the internet)

Intro to Distributed Programming

- You are probably very familiar with distributed programming as a user
- ext messaging systems, multiplayer video games, and social media apps
- You solicit services from another program (usually running on another computer)
- Dividing a problem into several tasks and writing a program for each task that communicates with the programs for other tasks is called *distributed programming*
- The tasks cooperate to solve a problem
- Each task defines a *component* which is a program
- Each component itself may be divided into subtasks and may be solved using one or more computers
- The components cooperate by communicating with each other using message-passing
- Messages are exchanged via a network (e.g., the internet)
- For messages to be exchanged a *communication protocol* must be designed
- A communication protocol defines the messages that may be exchanged and when messages are exchanged

Intro to Distributed Programming

- Messages cannot be arbitrary
- There are finite number of data types that are suitable for transmission
- For example, a number is suitable for transmission but a `posn` is unsuitable for transmission

Intro to Distributed Programming

- Messages cannot be arbitrary
- There are finite number of data types that are suitable for transmission
- For example, a number is suitable for transmission but a `posn` is unsuitable for transmission
- If a component needs to send data that is unsuitable for transmission the data must be marshaled
- *Marshalling* is the process of transforming data that is unsuitable for transmission into data that is suitable for transmission
- The component receiving marshaled data must unmarshal it
- *Unmarshalling* is the process of reconstructing the original data from marshaled data

Intro to Distributed Programming

- Messages cannot be arbitrary
- There are finite number of data types that are suitable for transmission
- For example, a number is suitable for transmission but a `posn` is unsuitable for transmission
- If a component needs to send data that is unsuitable for transmission the data must be marshaled
- *Marshalling* is the process of transforming data that is unsuitable for transmission into data that is suitable for transmission
- The component receiving marshaled data must unmarshal it
- *Unmarshalling* is the process of reconstructing the original data from marshaled data
- Marshalling and unmarshalling functions are inverses of each other
- `(unmarshal (marshal x)) = x`
- `(marshal (unmarshal message)) = message`

Intro to Distributed Programming

- A pervasively used distributed system architecture is the client-server architecture
- A server is a program that provides services or coordinates the cooperation among clients
- A client is a program that performs a task (usually) in cooperation with other clients to solve a problem

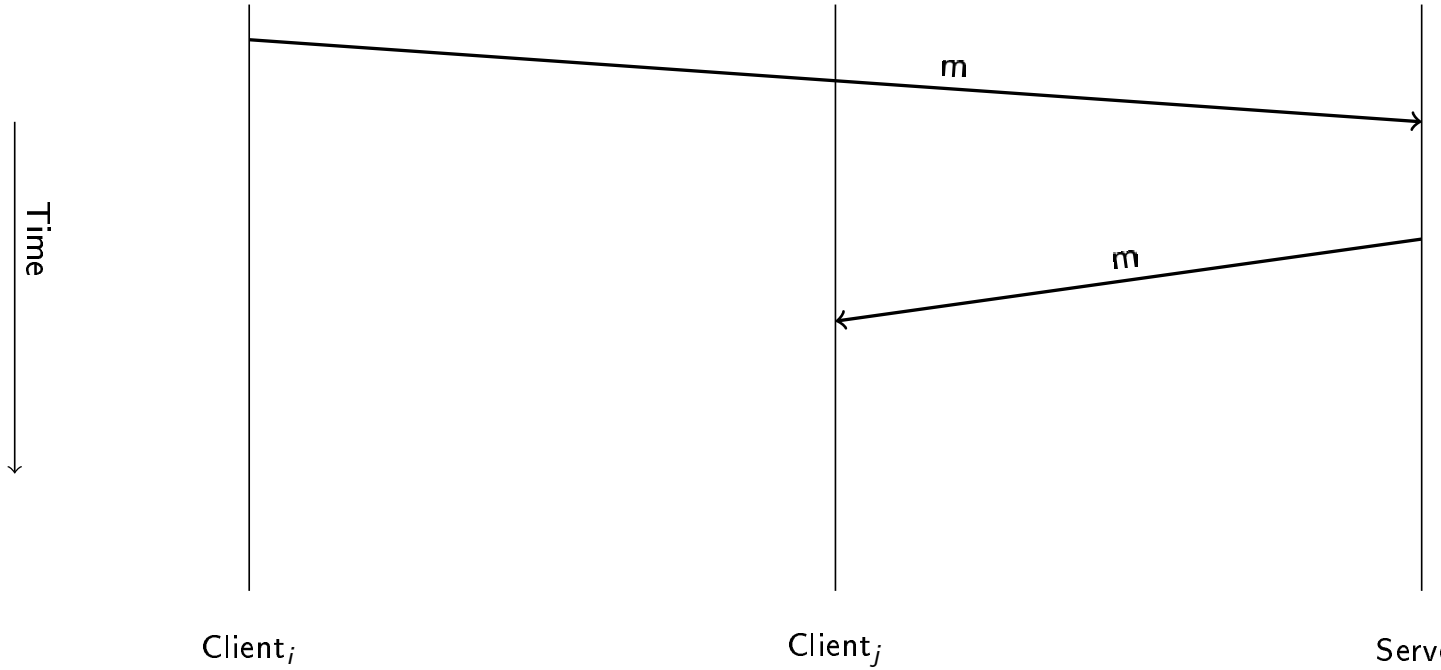
Intro to Distributed Programming

- A pervasively used distributed system architecture is the client-server architecture
- A server is a program that provides services or coordinates the cooperation among clients
- A client is a program that performs a task (usually) in cooperation with other clients to solve a problem
- A client on one computer requests services from the server that typically runs on another computer
- All communication between clients occurs through the server
- If $\texttt{Client}_i$ needs to send a message, `m`, to $\texttt{Client}_j$ then $\texttt{Client}_i$ sends `m` to the server and the server sends `m` to $\texttt{Client}_j$
- This `communication chain` is part of the communication protocol
- There can be many communication chains for different events

Intro to Distributed Programming

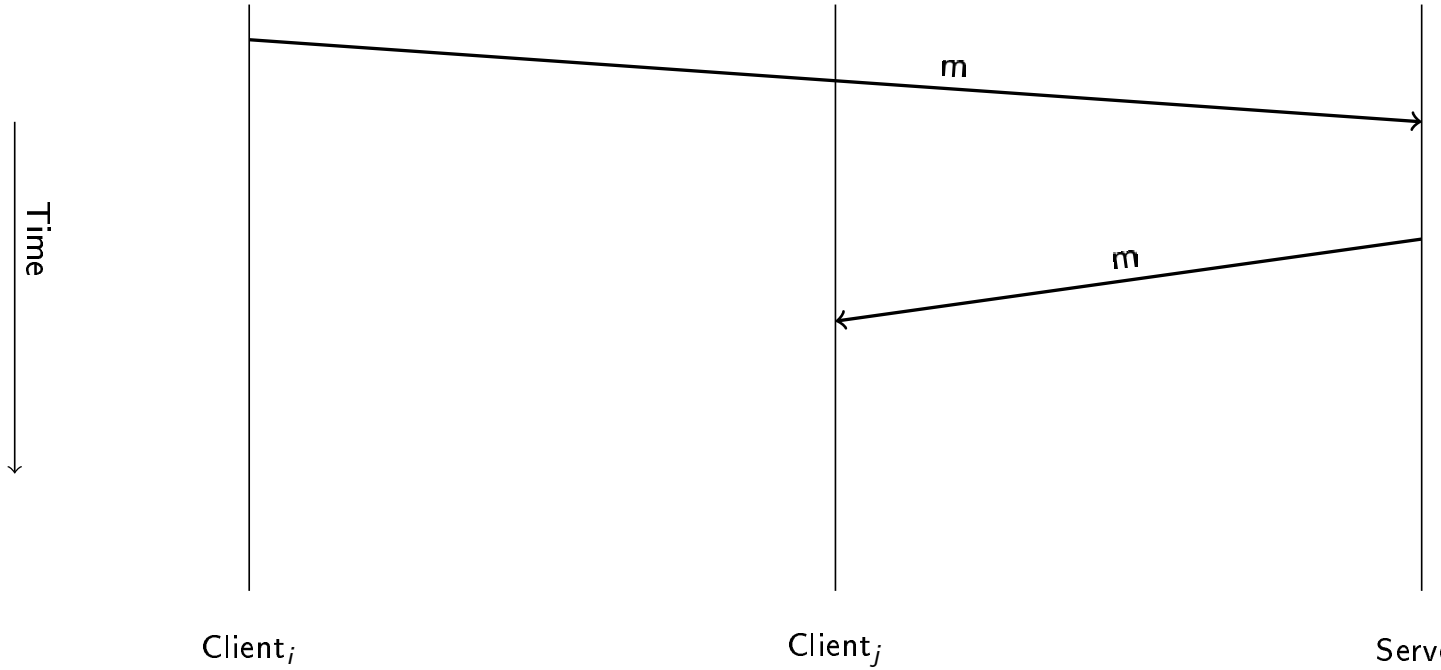
- A pervasively used distributed system architecture is the client-server architecture
- A server is a program that provides services or coordinates the cooperation among clients
- A client is a program that performs a task (usually) in cooperation with other clients to solve a problem
- A client on one computer requests services from the server that typically runs on another computer
- All communication between clients occurs through the server
- If $\texttt{Client}_i$ needs to send a message, `m`, to $\texttt{Client}_j$ then $\texttt{Client}_i$ sends `m` to the server and the server sends `m` to $\texttt{Client}_j$
- This `communication chain` is part of the communication protocol
- There can be many communication chains for different events
- A communication protocol may be specified using `protocol diagrams`
- In a protocol diagram the horizontal axis represents the components and the vertical access represents time (which grows from top to bottom).
- Messages are represented by solid arrows from source to destination at a slight angle
- The angle is used to emphasize that communication is not instantaneous
- Dashed arrows are used to represent communication that is implemented by the `API`

Intro to Distributed Programming



- `Client`$_i$ needs to send a message, `m`, to `Client`$_j$
- Communication chain: `Client`$_i$ sends `m` to the server and the server sends `m` to `Client`$_j$

Intro to Distributed Programming



- `Client`$_i$ needs to send a message, `m`, to `Client`$_j$
- Communication chain: `Client`$_i$ sends `m` to the server and the server sends `m` to `Client`$_j$
- A *thin server* is one that provides a minimal number of services like message broadcasting
- A *thick server* provides services that involve actual computing that directly contributes to solving a problem

Intro to Distributed Programming

A Design Recipe for Distributed Programming

- Distributed programming entails many characteristics that are not present when a program has a single task
- It is important to carefully design the different components and the communication protocol

Intro to Distributed Programming

A Design Recipe for Distributed Programming

- Distributed programming entails many characteristics that are not present when a program has a single task
- It is important to carefully design the different components and the communication protocol
- There is a design recipe to help guide your development
- Less prescriptive than the previously discussed design recipes

Intro to Distributed Programming

A Design Recipe for Distributed Programming

- Distributed programming entails many characteristics that are not present when a program has a single task
- It is important to carefully design the different components and the communication protocol
- There is a design recipe to help guide your development
- Less prescriptive than the previously discussed design recipes
- It does not tell you when to use a certain type of expression nor does it dictate what the parameters to a function must be
- It guides you through the development of a distributed program assuming that you have mastered the design recipes previously studied
- Each step still has a specific outcome

Intro to Distributed Programming

A Design Recipe for Distributed Programming

- The design recipe for distributed programming is:
  1. Divide the problem into components.
  2. Draft data definitions for the different components.
  3. Design a communication protocol.
  4. Design marshalling and unmarshalling functions.
  5. Design and implement the components.
  6. Test your program.

Intro to Distributed Programming

More on the Universe API

- The universe teachpack provides the functionality to develop distributed multiplayer games
- Each player and the server are components
- A player is executed using a `big-bang`-expression

Intro to Distributed Programming

More on the Universe API

- The universe teachpack provides the functionality to develop distributed multiplayer games
- Each player and the server are components
- A player is executed using a `big-bang`-expression
- A server manages a universe (e.g., a collection of players)
- Executed using a `universe`-expression

Intro to Distributed Programming

More on the Universe API

- The universe teachpack provides the functionality to develop distributed multiplayer games
- Each player and the server are components
- A player is executed using a `big-bang`-expression
- A server manages a universe (e.g., a collection of players)
- Executed using a `universe`-expression
- The players in a universe exchange messages with the server
- All communication occurs through the server

Intro to Distributed Programming

More on the Universe API

- The universe teachpack provides two functions to create messages:
- `make-package` is used by a client to create a structure that contains a (possibly new) world and a to-server message

Intro to Distributed Programming

More on the Universe API

- The universe teachpack provides two functions to create messages:
- `make-package` is used by a client to create a structure that contains a (possibly new) world and a to-server message
- `make-bundle` is used by the universe server to create a structure that contains a (possibly new) universe, a list of *mails* to any of the players, and a list of worlds to be disconnected from the universe
- Observe that a `bundle` contains an arbitrary number of mails and not an arbitrary number of to-world messages
- A mail is a structure, built using `make-mail`, that contains the recipient player and a `to-client message`

Intro to Distributed Programming

More on the Universe API

- The universe teachpack provides two functions to create messages:
- `make-package` is used by a client to create a structure that contains a (possibly new) world and a to-server message
- `make-bundle` is used by the universe server to create a structure that contains a (possibly new) universe, a list of *mails* to any of the players, and a list of worlds to be disconnected from the universe
- Observe that a `bundle` contains an arbitrary number of mails and not an arbitrary number of to-world messages
- A mail is a structure, built using `make-mail`, that contains the recipient player and a `to-client message`
- A message must be an `S-expression`:

```
A universe S-expression (sexpr) is either a:
  1. string
  2. symbol
  3. number
  4. Boolean
  5. character
  6 (listof sexpr)
```

- Nothing else is suitable for transmission in a universe program

Intro to Distributed Programming

More on the Universe API

- The universe teachpack provides two functions to create messages:
- `make-package` is used by a client to create a structure that contains a (possibly new) world and a to-server message
- `make-bundle` is used by the universe server to create a structure that contains a (possibly new) universe, a list of *mails* to any of the players, and a list of worlds to be disconnected from the universe
- Observe that a `bundle` contains an arbitrary number of mails and not an arbitrary number of to-world messages
- A mail is a structure, built using `make-mail`, that contains the recipient player and a `to-client message`
- A message must be an `S-expression`:

```
A universe S-expression (sexpr) is either a:
  1. string
  2. symbol
  3. number
  4. Boolean
  5. character
  6 (listof sexpr)
```

- Nothing else is suitable for transmission in a universe program
- Structures may not be transmitted
- If a structure must be transmitted you must implement marshalling and unmarshalling functions

Intro to Distributed Programming

More on the Universe API

- Players register with the server and have a handler to process messages

Intro to Distributed Programming

More on the Universe API

- Players register with the server and have a handler to process messages
- Need a string for the internet address of the computer running the server
- Your internet address is the value of ISL+'s `LOCALHOST` variable
- During development run all components using `LOCALHOST`

Intro to Distributed Programming

More on the Universe API

- Players register with the server and have a handler to process messages
- Need a string for the internet address of the computer running the server
- Your internet address is the value of ISL+'s LOCALHOST variable
- During development run all components using LOCALHOST
- For example, the run function for a player may look like this:

```
;; string → world   Purpose: To run the game
(define (run a-name)
  (big-bang
    INIT-WORLD
    (on-draw draw-world)
    (on-key process-key)
    (on-tick update-world)
    (stop-when game-over?)
    (register LOCALHOST)
    (on-receive process-message)
    (name a-name)))
```

- register clause: internet address for server
- If LOCALHOST then the server is running on your computer

Intro to Distributed Programming

More on the Universe API

- Players register with the server and have a handler to process messages
- Need a string for the internet address of the computer running the server
- Your internet address is the value of ISL+'s LOCALHOST variable
- During development run all components using LOCALHOST
- For example, the run function for a player may look like this:

```
;; string → world   Purpose: To run the game
(define (run a-name)
  (big-bang
    INIT-WORLD
    (on-draw draw-world)
    (on-key process-key)
    (on-tick update-world)
    (stop-when game-over?)
    (register LOCALHOST)
    (on-receive process-message)
    (name a-name)))
```

- register clause: internet address for server
- If LOCALHOST then the server is running on your computer
- It is noteworthy that handlers that may create a new world may return a world or a package
- A package is returned when a message is sent to the server
- Otherwise, it suffices to return a world
- The world in a package becomes the next world

Intro to Distributed Programming

More on the Universe API

- The syntax to run server specifies initial universe and event handlers
- Events: new message arrival, registration request, clock tick, etc.

Intro to Distributed Programming

More on the Universe API

- The syntax to run server specifies initial universe and event handlers
- Events: new message arrival, registration request, clock tick, etc.
- The `run-server` function for the `universe` may look like this:

```
;; Z → universe   Purpose: Run the universe server
(define (run-server a-z) dummy parameter
  (universe initU
            (on-new add-new-world)
            (on-msg process-message)
            (on-disconnect rm-world)
            (on-tick process-tick)))
```

- Initial universe and the `on-new` and `on-msg` clauses are required

Intro to Distributed Programming

More on the Universe API

- The syntax to run server specifies initial universe and event handlers
- Events: new message arrival, registration request, clock tick, etc.
- The `run-server` function for the `universe` may look like this:

```
;; Z → universe   Purpose: Run the universe server
(define (run-server a-z) dummy parameter
  (universe initU
            (on-new add-new-world)
            (on-msg process-message)
            (on-disconnect rm-world)
            (on-tick process-tick)))
```

- Initial universe and the `on-new` and `on-msg` clauses are required
- `universe` teachpack represents clients as `iworld` structures
- Only `iworld` characteristic we may access is its name using `iworld-name`

Intro to Distributed Programming

More on the Universe API

- The syntax to run server specifies initial universe and event handlers
- Events: new message arrival, registration request, clock tick, etc.
- The `run-server` function for the `universe` may look like this:

```
;; Z → universe   Purpose: Run the universe server
(define (run-server a-z) dummy parameter
  (universe initU
            (on-new add-new-world)
            (on-msg process-message)
            (on-disconnect rm-world)
            (on-tick process-tick)))
```

- Initial universe and the `on-new` and `on-msg` clauses are required
- `universe` teachpack represents clients as `iworld` structures
- Only `iworld` characteristic we may access is its name using `iworld-name`
- For testing: `iworld1`, `iworld2`, and `iworld3`
- The following are the signatures required by the `universe` API for the handlers specified in `run-server` above:

```
  add-new-world: universe iworld         → bundle or universe
process-message: universe iworld message → bundle or universe
       rm-world: universe iworld         → bundle or universe
   process-tick: universe                → bundle or universe
```

- A `bundle` is returned when there is at least one `mail` to be sent to a player
- The `universe` in the bundle becomes the next `universe`
- If no mail needs to be sent then it suffices for a handler to return a universe

Intro to Distributed Programming

A Chat Application

- Let's develop of a chat tool

Intro to Distributed Programming

A Chat Application

- Let's develop of a chat tool
- Users share messages with everyone that is connected to the server
- A user types a string of at most length 20 and sends it to the group of connected users
- The server receives a message from a user and broadcasts it to the rest of the users

Intro to Distributed Programming

A Chat Application

- Let's develop of a chat tool
- Users share messages with everyone that is connected to the server
- A user types a string of at most length 20 and sends it to the group of connected users
- The server receives a message from a user and broadcasts it to the rest of the users
- Two components: the chat client and the chat server
- Client draws the latest four messages and the message partially typed

Intro to Distributed Programming

A Chat Application

- Let's develop of a chat tool
- Users share messages with everyone that is connected to the server
- A user types a string of at most length 20 and sends it to the group of connected users
- The server receives a message from a user and broadcasts it to the rest of the users
- Two components: the chat client and the chat server
- Client draws the latest four messages and the message partially typed
- The client is also responsible for processing keystrokes
- `Enter` and new message is not empty: add as last message received and send to server

Intro to Distributed Programming

A Chat Application

- Let's develop of a chat tool
- Users share messages with everyone that is connected to the server
- A user types a string of at most length 20 and sends it to the group of connected users
- The server receives a message from a user and broadcasts it to the rest of the users
- Two components: the chat client and the chat server
- Client draws the latest four messages and the message partially typed
- The client is also responsible for processing keystrokes
- `Enter` and new message is not empty: add as last message received and send to server
- `Backspace` and the partially written message is not empty: last character is deleted

Intro to Distributed Programming

A Chat Application

- Let's develop of a chat tool
- Users share messages with everyone that is connected to the server
- A user types a string of at most length 20 and sends it to the group of connected users
- The server receives a message from a user and broadcasts it to the rest of the users
- Two components: the chat client and the chat server
- Client draws the latest four messages and the message partially typed
- The client is also responsible for processing keystrokes
- `Enter` and new message is not empty: add as last message received and send to server
- `Backspace` and the partially written message is not empty: last character is deleted
- `Shift` and `Tab` keys are ignored

Intro to Distributed Programming

A Chat Application

- Let's develop of a chat tool
- Users share messages with everyone that is connected to the server
- A user types a string of at most length 20 and sends it to the group of connected users
- The server receives a message from a user and broadcasts it to the rest of the users
- Two components: the chat client and the chat server
- Client draws the latest four messages and the message partially typed
- The client is also responsible for processing keystrokes
- `Enter` and new message is not empty: add as last message received and send to server
- `Backspace` and the partially written message is not empty: last character is deleted
- `Shift` and `Tab` keys are ignored
- Any other key is added to the message as long as the length of the new message is at most 20

Intro to Distributed Programming

A Chat Application

- The server is responsible for adding new users to the universe
- A new user is added only if her name is different from the name of every other user
- When a new user is added a message is sent to all users informing them of the name of the new user

Intro to Distributed Programming

A Chat Application

- The server is responsible for adding new users to the universe
- A new user is added only if her name is different from the name of every other user
- When a new user is added a message is sent to all users informing them of the name of the new user
- The server is also responsible for processing the messages
- For each message a new `mail` is created for every user except the sender of the message

Intro to Distributed Programming

A Chat Application

- Client program: data definition for a text message and a world are needed

Intro to Distributed Programming

A Chat Application

- Client program: data definition for a text message and a world are needed
- ```
  (define MAX-TM-LEN 20)
  ;; An text message (tm) is a string of length <= MAX-TM-LEN of
  ;; keystrokes that does not contain a return, a backspace, a shift,
  ;; or a tab.
  ```
  ```

Intro to Distributed Programming

A Chat Application

- Client program: data definition for a text message and a world are needed
- ```
  (define MAX-TM-LEN 20)
  ;; An text message (tm) is a string of length <= MAX-TM-LEN of
  ;; keystrokes that does not contain a return, a backspace, a shift,
  ;; or a tab.
  ```
- ```
  ;; A world is a structure: (make-world tm tm tm tm tm)
  ;; that contains 5 text messages from left to right:
  ;; partially written to fourth most recent

  (define-struct world (tm1 tm2 tm3 tm4 tm5))

  ;; Sample worlds
  (define INIT-WORLD (make-world "" "" "" "" ""))
  (define A-WORLD
   (make-world "Wanna hang?" "Good thnx" "Good and you?"
               "Hi, how are you" "Hi"))
   (define B-WORLD (make-world "12345678901234567890"
                               "Guess a number" "" "" ""))
  ```

Intro to Distributed Programming

A Chat Application

- ```
  ;; world ... → ...
  ;; Purpose:
  ;; (define (f-on-world a-world)
  ;;   (...(world-tm1 a-world)...(world-tm2 a-world)
  ;;    ...(world-tm3 a-world)...(world-tm4 a-world)
  ;;    ...(world-tm5 a-world)...))

  ;; Sample worlds
  ;; (define WORLD-0 (make-world ... ... ... ... ...)) ...

  ;; Sample expressions for f-on-world
  ;; (define WORLD-0-VAL ... WORLD-0 ...)

  ;; Tests using sample computations for f-on-world
  ;; (check-expect (f-on-world WORLD-0 ...) WORLD-0-VAL) ...

  ;;  Tests using sample values for f-on-world
  ;;  (check-expect (f-on-world (make-world ...) ...) ...) ...
  ```
  ```

Intro to Distributed Programming

A Chat Application

- The server needs to track the worlds in the universe

Intro to Distributed Programming

A Chat Application

- The server needs to track the worlds in the universe
-
```
;; A universe is a (listof iworld)
;; universe ... → ...   Purpose:
;; (define (f-on-universe a-universe)
;;   (if (empty? a-universe)
;;       ...
;;       (... (first a-universe) ...
;;        ... (f-on-universe (rest a-universe)) ...)))
;; Sample universes
;; (define UNIV-0 '())   (define UNIV-1 ...) ...
;;
;; Sample expressions for f-on-universe
;; (define UNIV-0-VAL ... UNIV-0 ...)
;; (define UNIV-1-VAL ... UNIV-1 ...)
;;
;; Tests using sample computations for f-on-universe
;; (check-expect (f-on-universe UNIV-0 ...) UNIV-0-VAL)
;; (check-expect (f-on-universe UNIV-1 ...) UNIV-1-VAL)...
;; Tests using sample values for f-on-universe
;; (check-expect (f-on-universe ... ...) ...) ...
;; ;; Sample universes
(define INIT-UNIV '())
(define A-UNIV (list iworld1 iworld2))
```

Intro to Distributed Programming

A Chat Application

- A communication chain is sparked by the keystroke `Enter`
- Think carefully about what ought to happen when $\texttt{Client}_i$ sends a message

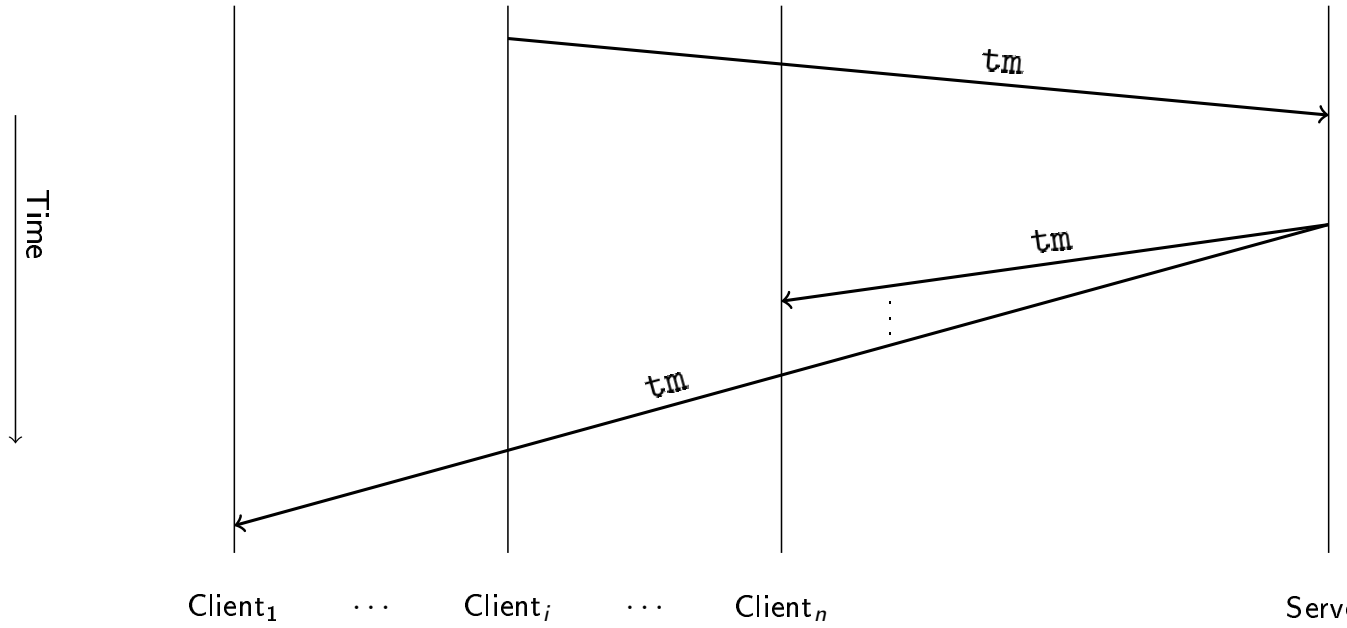
Intro to Distributed Programming

A Chat Application

- A communication chain is sparked by the keystroke `Enter`
- Think carefully about what ought to happen when $\texttt{Client}_i$ sends a message
- The partially written message is complete and must be sent to the server
- Upon receiving this message the server must broadcast it the other users

Intro to Distributed Programming

A Chat Application



- A communication chain is sparked by the keystroke `Enter`
- Think carefully about what ought to happen when `Client`$_i$ sends a message
- The partially written message is complete and must be sent to the server
- Upon receiving this message the server must broadcast it the other users
- Solid arrow from `Client`$_i$ to `Server`
- `Client`$_i$ may need to marshal its message
- `Server` might have to unmarshal the message
- Not the case here because `tm` is a string

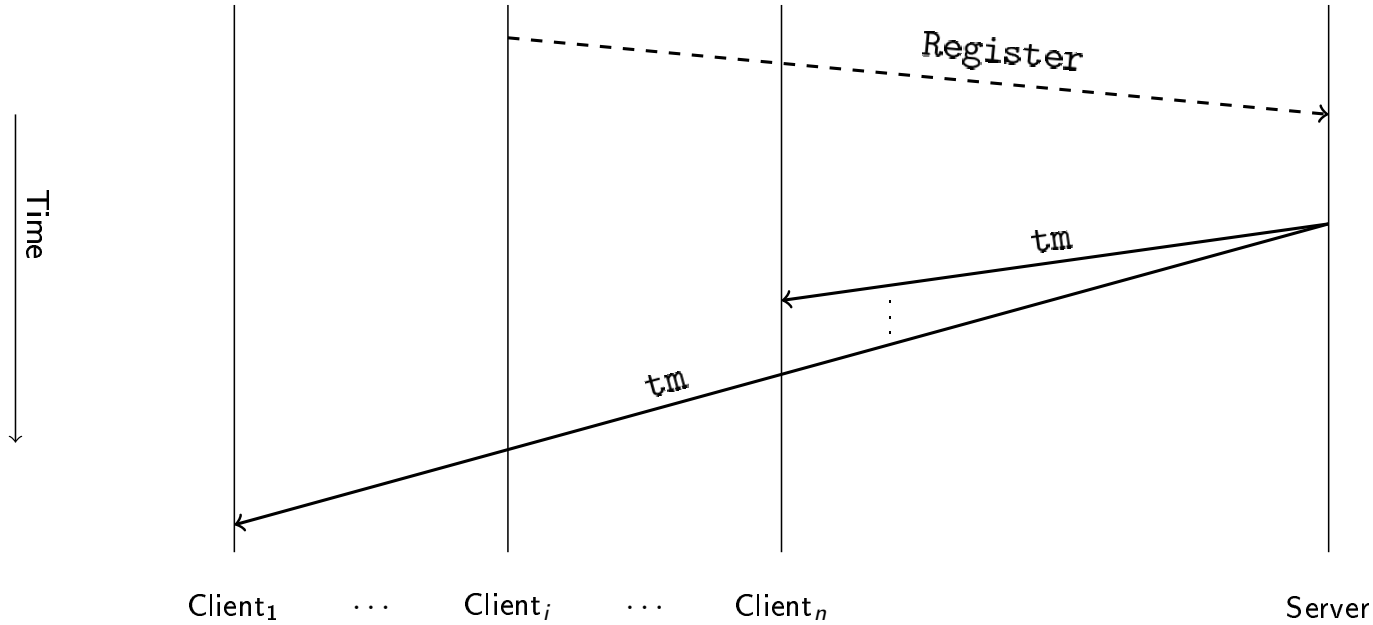
Intro to Distributed Programming

A Chat Application

- Communication chain is also sparked when a new client joins the server
- Server must broadcast a message to all the clients that a new user has joined the chat
- No marshalling or unmarshalling is necessary

Intro to Distributed Programming

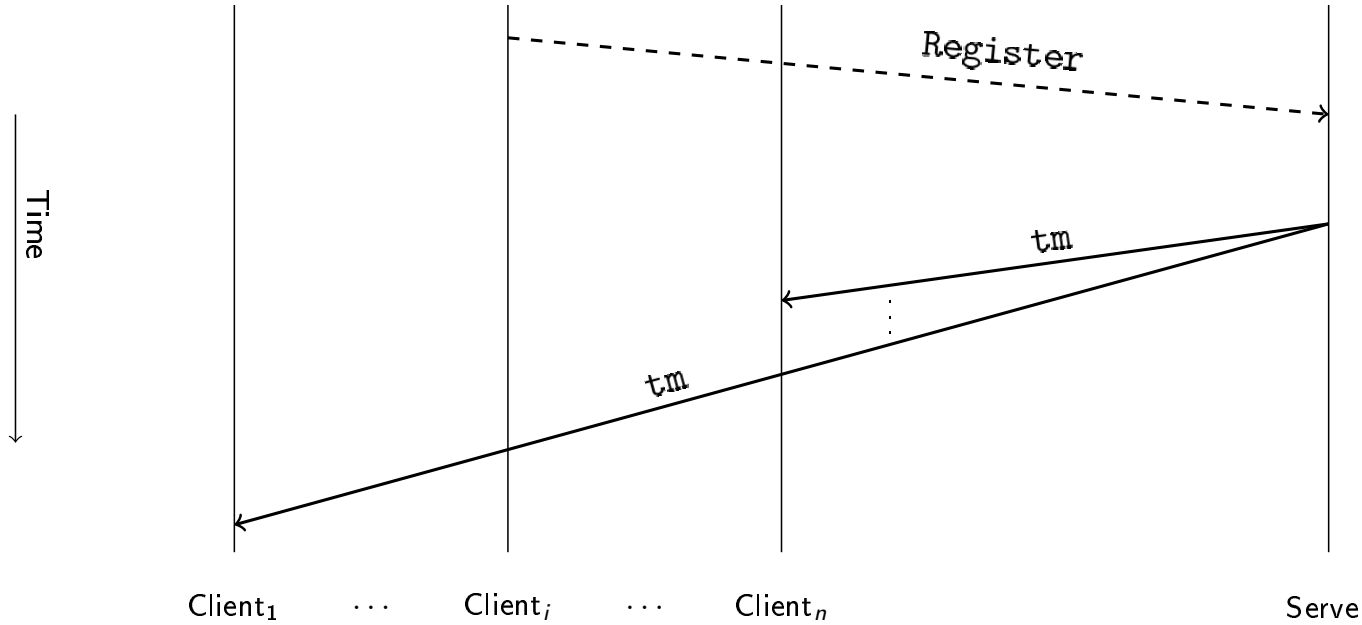
A Chat Application



- Communication chain is also sparked when a new client joins the server
- Server must broadcast a message to all the clients that a new user has joined the chat
- No marshalling or unmarshalling is necessary

Intro to Distributed Programming

A Chat Application



- Communication chain is also sparked when a new client joins the server
- Server must broadcast a message to all the clients that a new user has joined the chat
- No marshalling or unmarshalling is necessary
- We can now define `to-server` and `to-client` messages:

```
;; A to-server message is a tm

;; A to-client message is a tm
```

Intro to Distributed Programming

Client

- `draw-world`: draw the four most recent `tms` and the partially written `tm`
- Rendered by drawing the four most recent `tms` above each other with the least recent on the top and most recent on the bottom
- Red to visually separate the partially written `tm`
- No messages need to be sent to the server when the world is rendered

Intro to Distributed Programming

Client

- `draw-world`: draw the four most recent `tms` and the partially written `tm`
- Rendered by drawing the four most recent `tms` above each other with the least recent on the top and most recent on the bottom
- Red to visually separate the partially written `tm`
- No messages need to be sent to the server when the world is rendered
- `;; world → image  Purpose: To draw the given world`
  `(define (draw-world a-world)`
  `  (draw-world a-world)))`

Intro to Distributed Programming

Client

- `draw-world`: draw the four most recent `tms` and the partially written `tm`
- Rendered by drawing the four most recent `tms` above each other with the least recent on the top and most recent on the bottom
- Red to visually separate the partially written `tm`
- No messages need to be sent to the server when the world is rendered
- `;; world → image  Purpose: To draw the given world`
  `(define (draw-world a-world)`
- `(local [(define WIDTH 270) (define HEIGHT 170) (define VSPACE 10)`
  `(define E-SCENE (empty-scene WIDTH HEIGHT))`

- `(draw-world a-world)))`

Intro to Distributed Programming

Client

- `draw-world`: draw the four most recent `tms` and the partially written `tm`
- Rendered by drawing the four most recent `tms` above each other with the least recent on the top and most recent on the bottom
- Red to visually separate the partially written `tm`
- No messages need to be sent to the server when the world is rendered
-
```
;; world → image  Purpose: To draw the given world
(define (draw-world a-world)
```
-
```
 (local [(define WIDTH 270) (define HEIGHT 170) (define VSPACE 10)
         (define E-SCENE (empty-scene WIDTH HEIGHT))
```
-
```
         ;; world → image  Purpose: To draw the given world
         (define (draw-world w)
          (local [(define IMG5 (make-tm-img (world-tm5 w)))
                  (define IMG4 (make-tm-img (world-tm4 w)))
                  (define IMG3 (make-tm-img (world-tm3 w)))
                  (define IMG2 (make-tm-img (world-tm2 w)))
                  (define IMG1 (make-tm-img (world-tm1 w)))]
            (add-line (place-tm IMG5 1
                                (place-tm IMG4 2
                                  (place-tm IMG3 3
                                    (place-tm IMG2 4 (place-tm IMG1 6 E-SCENE)))))
             0 (* 5 (+ VSPACE (/ (image-height IMG2) 2)))
             (sub1 WIDTH) (* 5 (+ VSPACE (/ (image-height IMG2) 2)))
             "red"))])
```
-
```
  (draw-world a-world)))
```

Intro to Distributed Programming

Client

- `draw-world`: draw the four most recent `tms` and the partially written `tm`
- Rendered by drawing the four most recent `tms` above each other with the least recent on the top and most recent on the bottom
- Red to visually separate the partially written `tm`
- No messages need to be sent to the server when the world is rendered
-
```
;; world → image  Purpose: To draw the given world
(define (draw-world a-world)
```
-
```
 (local [(define WIDTH 270) (define HEIGHT 170) (define VSPACE 10)
         (define E-SCENE (empty-scene WIDTH HEIGHT))
```
-
```
         ;; tm → image  Purpose: Convert the given text to an image
         (define (make-tm-img a-tm) (text a-tm 24 "brown"))
```
-
```
         ;; world → image  Purpose: To draw the given world
         (define (draw-world w)
          (local [(define IMG5 (make-tm-img (world-tm5 w)))
                  (define IMG4 (make-tm-img (world-tm4 w)))
                  (define IMG3 (make-tm-img (world-tm3 w)))
                  (define IMG2 (make-tm-img (world-tm2 w)))
                  (define IMG1 (make-tm-img (world-tm1 w)))]
            (add-line (place-tm IMG5 1
                                (place-tm IMG4 2
                                  (place-tm IMG3 3
                                    (place-tm IMG2 4 (place-tm IMG1 6 E-SCENE)))))
             0 (* 5 (+ VSPACE (/ (image-height IMG2) 2)))
             (sub1 WIDTH) (* 5 (+ VSPACE (/ (image-height IMG2) 2)))
             "red")))]
```
-
```
  (draw-world a-world)))
```

Intro to Distributed Programming

Client

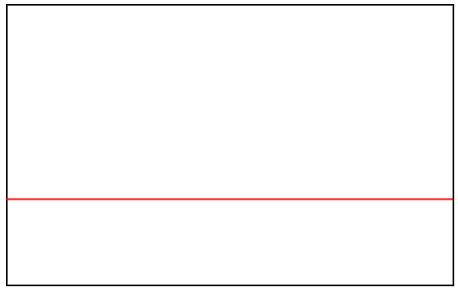
- `draw-world`: draw the four most recent `tm`s and the partially written `tm`
- Rendered by drawing the four most recent `tm`s above each other with the least recent on the top and most recent on the bottom
- Red to visually separate the partially written `tm`
- No messages need to be sent to the server when the world is rendered
-
```
;; world → image  Purpose: To draw the given world
(define (draw-world a-world)
```
-
```
 (local [(define WIDTH 270) (define HEIGHT 170) (define VSPACE 10)
         (define E-SCENE (empty-scene WIDTH HEIGHT))
```
-
```
         ;; tm → image  Purpose: Convert the given text to an image
         (define (make-tm-img a-tm) (text a-tm 24 "brown"))
```
-
```
         ;;image natnum>0 image → image  Purpose:Place given tm image
         (define (place-tm img factor scn)
          (place-image img
                       (add1 (/ (image-width img) 2))
                       (* factor (+ VSPACE (/ (image-height img) 2)))
                       scn))
```
-
```
         ;; world → image  Purpose: To draw the given world
         (define (draw-world w)
          (local [(define IMG5 (make-tm-img (world-tm5 w)))
                  (define IMG4 (make-tm-img (world-tm4 w)))
                  (define IMG3 (make-tm-img (world-tm3 w)))
                  (define IMG2 (make-tm-img (world-tm2 w)))
                  (define IMG1 (make-tm-img (world-tm1 w)))]
            (add-line (place-tm IMG5 1
                                (place-tm IMG4 2
                                  (place-tm IMG3 3
                                    (place-tm IMG2 4 (place-tm IMG1 6 E-SCENE)))))
              0 (* 5 (+ VSPACE (/ (image-height IMG2) 2)))
              (sub1 WIDTH) (* 5 (+ VSPACE (/ (image-height IMG2) 2)))
              "red"))])
```
-
```
  (draw-world a-world)))
```

Intro to Distributed Programming

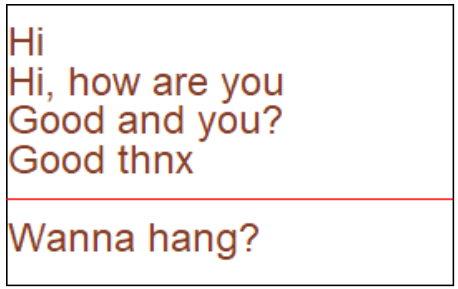
Client

- ;; Tests using sample values for draw-world

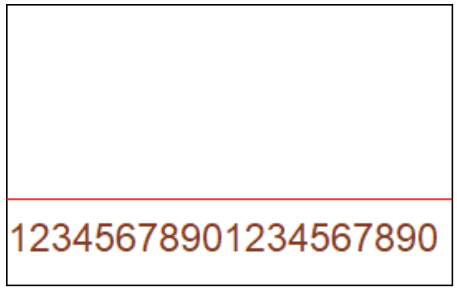
```
(check-expect (draw-world INIT-WORLD) )
```



```
(check-expect (draw-world A-WORLD) )
```




```
(check-expect (draw-world (make-world "12345678901234567890"
                                      "" "" "" ""))
```



)

Intro to Distributed Programming

Client

- If the given key is `Enter` and the partially written `tm` is empty the keystroke is ignored
- Otherwise, following the protocol, a package is created that sends the partially written `tm` to the server and updates the most recently `tms`

Intro to Distributed Programming

Client

- If the given key is `Enter` and the partially written `tm` is empty the keystroke is ignored
- Otherwise, following the protocol, a package is created that sends the partially written `tm` to the server and updates the most recently `tms`
- If the given key is `Backspace` the keystroke is ignored if the partially written `tm` is the empty string
- Otherwise, the last character of the partially written `tm` is removed

Intro to Distributed Programming

Client

- If the given key is `Enter` and the partially written `tm` is empty the keystroke is ignored
- Otherwise, following the protocol, a package is created that sends the partially written `tm` to the server and updates the most recently `tms`
- If the given key is `Backspace` the keystroke is ignored if the partially written `tm` is the empty string
- Otherwise, the last character of the partially written `tm` is removed
- If the given key is either `"Shift"` or `Tab` the keystroke is ignored

Intro to Distributed Programming

Client

- If the given key is `Enter` and the partially written `tm` is empty the keystroke is ignored
- Otherwise, following the protocol, a package is created that sends the partially written `tm` to the server and updates the most recently `tms`
- If the given key is `Backspace` the keystroke is ignored if the partially written `tm` is the empty string
- Otherwise, the last character of the partially written `tm` is removed
- If the given key is either `"Shift"` or `Tab` the keystroke is ignored
- Any other keystroke is added to the partially written `tm` if its length is less than or equal to `MAX-TM-LEN` and ignored otherwise

Intro to Distributed Programming

A Chat Application

- `;; world key → world or package  Purpose: Return next world after keystroke`
  `(define (process-key a-world a-key)`

Intro to Distributed Programming

A Chat Application

- ```
  ;; world key → world or package  Purpose: Return next world after keystroke
  (define (process-key a-world a-key)
  ```

- ```
  ;; Tests using sample computations for process-key
  (check-expect (process-key INIT-WORLD "\r") INITW-RET)
  (check-expect (process-key A-WORLD "\r")    AW-RET)
  (check-expect (process-key INIT-WORLD "\b") INITW-BACK)
  (check-expect (process-key A-WORLD "\b")    AW-BACK)
  (check-expect (process-key A-WORLD "shift") AW-SHIFT)
  (check-expect (process-key A-WORLD "\t")    AW-TAB)
  (check-expect (process-key B-WORLD "M")     BW-M)
  (check-expect (process-key INIT-WORLD "A")  INITW-A)
  ;; Tests using sample values for process-key
  (check-expect (process-key (make-world "" "B" "A" "" "") "C")
                (make-world "C" "B" "A" "" ""))
  (check-expect (process-key (make-world "" "B" "A" "" "") "\r")
                (make-world "" "B" "A" "" ""))
  (check-expect (process-key (make-world "C" "B" "A" "" "") "\b")
                (make-world "" "B" "A" "" ""))
  ```

Intro to Distributed Programming

A Chat Application

```
;; world key → world or package  Purpose: Return next world after keystroke
(define (process-key a-world a-key)
  (local [(define tm1 (world-tm1 a-world))  (define tm2 (world-tm2 a-world))
          (define tm3 (world-tm3 a-world))  (define tm4 (world-tm4 a-world))
          (define tm5 (world-tm5 a-world))]
    (cond [(string=? a-key "\r")
           (if (string=? tm1 "") a-world
               (make-package (make-world "" tm1 tm2 tm3 tm4) tm1))]
```

```
;; Tests using sample computations for process-key
(check-expect (process-key INIT-WORLD "\r") INITW-RET)
(check-expect (process-key A-WORLD "\r")    AW-RET)
(check-expect (process-key INIT-WORLD "\b") INITW-BACK)
(check-expect (process-key A-WORLD "\b")    AW-BACK)
(check-expect (process-key A-WORLD "shift") AW-SHIFT)
(check-expect (process-key A-WORLD "\t")    AW-TAB)
(check-expect (process-key B-WORLD "M")     BW-M)
(check-expect (process-key INIT-WORLD "A")  INITW-A)
;; Tests using sample values for process-key
(check-expect (process-key (make-world "" "B" "A" "" "") "C")
              (make-world "C" "B" "A" "" ""))
(check-expect (process-key (make-world "" "B" "A" "" "") "\r")
              (make-world "" "B" "A" "" ""))
(check-expect (process-key (make-world "C" "B" "A" "" "") "\b")
              (make-world "" "B" "A" "" ""))
```

Intro to Distributed Programming

A Chat Application

```
;; world key → world or package  Purpose: Return next world after keystroke
(define (process-key a-world a-key)
  (local [(define tm1 (world-tm1 a-world))  (define tm2 (world-tm2 a-world))
          (define tm3 (world-tm3 a-world))  (define tm4 (world-tm4 a-world))
          (define tm5 (world-tm5 a-world))]
    (cond [(string=? a-key "\r")
           (if (string=? tm1 "") a-world
               (make-package (make-world "" tm1 tm2 tm3 tm4) tm1))]
          [(string=? a-key "\b")
           (if (string=? tm1 "") a-world
               (make-world
                (substring tm1 0 (sub1 (string-length tm1))) tm2 tm3 tm4 tm5))]
```

```
;; Tests using sample computations for process-key
(check-expect (process-key INIT-WORLD "\r") INITW-RET)
(check-expect (process-key A-WORLD "\r")    AW-RET)
(check-expect (process-key INIT-WORLD "\b") INITW-BACK)
(check-expect (process-key A-WORLD "\b")    AW-BACK)
(check-expect (process-key A-WORLD "shift") AW-SHIFT)
(check-expect (process-key A-WORLD "\t")    AW-TAB)
(check-expect (process-key B-WORLD "M")     BW-M)
(check-expect (process-key INIT-WORLD "A")  INITW-A)
;; Tests using sample values for process-key
(check-expect (process-key (make-world "" "B" "A" "" "") "C")
              (make-world "C" "B" "A" "" ""))
(check-expect (process-key (make-world "" "B" "A" "" "") "\r")
              (make-world "" "B" "A" "" ""))
(check-expect (process-key (make-world "C" "B" "A" "" "") "\b")
              (make-world "" "B" "A" "" ""))
```

Intro to Distributed Programming

A Chat Application

```
;; world key → world or package  Purpose: Return next world after keystroke
(define (process-key a-world a-key)
  (local [(define tm1 (world-tm1 a-world))  (define tm2 (world-tm2 a-world))
          (define tm3 (world-tm3 a-world))  (define tm4 (world-tm4 a-world))
          (define tm5 (world-tm5 a-world))]
    (cond [(string=? a-key "\r")
           (if (string=? tm1 "") a-world
               (make-package (make-world "" tm1 tm2 tm3 tm4) tm1))]
          [(string=? a-key "\b")
           (if (string=? tm1 "") a-world
               (make-world
                (substring tm1 0 (sub1 (string-length tm1))) tm2 tm3 tm4 tm5))]
          [(or (string=? a-key "shift") (string=? a-key "\t")) a-world]
```

```
;; Tests using sample computations for process-key
(check-expect (process-key INIT-WORLD "\r") INITW-RET)
(check-expect (process-key A-WORLD "\r")    AW-RET)
(check-expect (process-key INIT-WORLD "\b") INITW-BACK)
(check-expect (process-key A-WORLD "\b")    AW-BACK)
(check-expect (process-key A-WORLD "shift") AW-SHIFT)
(check-expect (process-key A-WORLD "\t")    AW-TAB)
(check-expect (process-key B-WORLD "M")     BW-M)
(check-expect (process-key INIT-WORLD "A")  INITW-A)
;; Tests using sample values for process-key
(check-expect (process-key (make-world "" "B" "A" "" "") "C")
              (make-world "C" "B" "A" "" ""))
(check-expect (process-key (make-world "" "B" "A" "" "") "\r")
              (make-world "" "B" "A" "" ""))
(check-expect (process-key (make-world "C" "B" "A" "" "") "\b")
              (make-world "" "B" "A" "" ""))
```

Intro to Distributed Programming

A Chat Application

```
;; world key → world or package  Purpose: Return next world after keystroke
(define (process-key a-world a-key)
  (local [(define tm1 (world-tm1 a-world))  (define tm2 (world-tm2 a-world))
          (define tm3 (world-tm3 a-world))  (define tm4 (world-tm4 a-world))
          (define tm5 (world-tm5 a-world))]
    (cond [(string=? a-key "\r")
           (if (string=? tm1 "") a-world
               (make-package (make-world "" tm1 tm2 tm3 tm4) tm1))]
          [(string=? a-key "\b")
           (if (string=? tm1 "") a-world
               (make-world
                (substring tm1 0 (sub1 (string-length tm1))) tm2 tm3 tm4 tm5))]
          [(or (string=? a-key "shift") (string=? a-key "\t")) a-world]
          [else (if (= (string-length tm1) MAX-TM-LEN) a-world
                    (make-world (string-append tm1 a-key) tm2 tm3 tm4 tm5))])))
;; Tests using sample computations for process-key
(check-expect (process-key INIT-WORLD "\r") INITW-RET)
(check-expect (process-key A-WORLD "\r")    AW-RET)
(check-expect (process-key INIT-WORLD "\b") INITW-BACK)
(check-expect (process-key A-WORLD "\b")    AW-BACK)
(check-expect (process-key A-WORLD "shift") AW-SHIFT)
(check-expect (process-key A-WORLD "\t")    AW-TAB)
(check-expect (process-key B-WORLD "M")     BW-M)
(check-expect (process-key INIT-WORLD "A")  INITW-A)
;; Tests using sample values for process-key
(check-expect (process-key (make-world "" "B" "A" "" "") "C")
              (make-world "C" "B" "A" "" ""))
(check-expect (process-key (make-world "" "B" "A" "" "") "\r")
              (make-world "" "B" "A" "" ""))
(check-expect (process-key (make-world "C" "B" "A" "" "") "\b")
              (make-world "" "B" "A" "" ""))
```

Intro to Distributed Programming

A Chat Application

- The protocol diagrams inform us that the only message that a client receives is a `tm`
- The received `tm` becomes the most recently received message and the other messages are moved to the next oldest slots
- The fourth most recent message is discarded to make room for the new `tm`

Intro to Distributed Programming

A Chat Application

```
;; Sample expressions for process-message
(define IWM (make-world (world-tm1 INIT-WORLD)
                        "It's me"
                        (world-tm2 INIT-WORLD)
                        (world-tm3 INIT-WORLD)
                        (world-tm4 INIT-WORLD)))
(define BWM (make-world (world-tm1 B-WORLD)
                        "I think so!"
                        (world-tm2 B-WORLD)
                        (world-tm3 B-WORLD)
                        (world-tm4 B-WORLD)))
```

Intro to Distributed Programming

A Chat Application

- ```
  ;; world message → world
  ;; Purpose: Process the given message
  (define (process-message a-world a-message)
  ```

- ```
  ;; Sample expressions for process-message
  (define IWM (make-world (world-tm1 INIT-WORLD)
                          "It's me"
                          (world-tm2 INIT-WORLD)
                          (world-tm3 INIT-WORLD)
                          (world-tm4 INIT-WORLD)))
  (define BWM (make-world (world-tm1 B-WORLD)
                          "I think so!"
                          (world-tm2 B-WORLD)
                          (world-tm3 B-WORLD)
                          (world-tm4 B-WORLD)))
  ```

Intro to Distributed Programming

A Chat Application

- ```
  ;; world message → world
  ;; Purpose: Process the given message
  (define (process-message a-world a-message)
  ```

- ```
  ;; Sample expressions for process-message
  (define IWM (make-world (world-tm1 INIT-WORLD)
                          "It's me"
                          (world-tm2 INIT-WORLD)
                          (world-tm3 INIT-WORLD)
                          (world-tm4 INIT-WORLD)))
  (define BWM (make-world (world-tm1 B-WORLD)
                          "I think so!"
                          (world-tm2 B-WORLD)
                          (world-tm3 B-WORLD)
                          (world-tm4 B-WORLD)))
  ```

- ```
  ;; Tests using sample computations and values for process-message
  (check-expect (process-message INIT-WORLD "It's me")  IWM)
  (check-expect (process-message B-WORLD "I think so!") BWM)
  ;; Tests using sample computations for process-message
  (check-expect
    (process-message (make-world "What d" "You and I?" "" "" "") "No way")
    (make-world "What d" "No way" "You and I?" "" ""))
  ```
  ```

Intro to Distributed Programming

A Chat Application

- ```
  ;; world message → world
  ;; Purpose: Process the given message
  (define (process-message a-world a-message)
  ```
- ```
    (make-world (world-tm1 a-world)
                a-message
                (world-tm2 a-world)
                (world-tm3 a-world)
                (world-tm4 a-world)))
  ```
- ```
  ;; Sample expressions for process-message
  (define IWM (make-world (world-tm1 INIT-WORLD)
                          "It's me"
                          (world-tm2 INIT-WORLD)
                          (world-tm3 INIT-WORLD)
                          (world-tm4 INIT-WORLD)))
  (define BWM (make-world (world-tm1 B-WORLD)
                          "I think so!"
                          (world-tm2 B-WORLD)
                          (world-tm3 B-WORLD)
                          (world-tm4 B-WORLD)))
  ```
- ```
  ;; Tests using sample computations and values for process-message
  (check-expect (process-message INIT-WORLD "It's me")  IWM)
  (check-expect (process-message B-WORLD "I think so!") BWM)
  ;; Tests using sample computations for process-message
  (check-expect
    (process-message (make-world "What d" "You and I?" "" "" "") "No way")
    (make-world "What d" "No way" "You and I?" "" ""))
  ```

Intro to Distributed Programming

Server

- Server must process new clients attempting to register and must process messages

Intro to Distributed Programming

Server

- Server must process new clients attempting to register and must process messages
- To add a new `iworld` to the server the name of the incoming `iworld` must be different from the name of any `iworld` already in the server

Intro to Distributed Programming

Server

- Server must process new clients attempting to register and must process messages
- To add a new `iworld` to the server the name of the incoming `iworld` must be different from the name of any `iworld` already in the server
- `map` is used to create a `mail` for each `iworld` in the `universe`

Intro to Distributed Programming

Server

- Server must process new clients attempting to register and must process messages
- To add a new `iworld` to the server the name of the incoming `iworld` must be different from the name of any `iworld` already in the server
- `map` is used to create a `mail` for each `iworld` in the `universe`
- No worlds are disconnected from the universe

Intro to Distributed Programming

Server

```
;; Sample expressions for add-new-world
(define ADD-INITU
        (make-bundle (list iworld1)
                     (map
                      (λ (iw)
                       (make-mail iw (string-append (iworld-name iworld3)
                                                    " has joined")))
                      INIT-UNIV)
                     '()))
(define ADD-AUNIV
       (make-bundle (list iworld3 iworld1 iworld2)
                    (map
                     (λ (iw)
                      (make-mail iw (string-append (iworld-name iworld3)
                                                   " has joined")))
                     A-UNIV)
                    '()))

(define ADD-REPEAT A-UNIV)

;; Tests using sample computations for add-new-world
(check-expect (add-new-world INIT-UNIV iworld1) ADD-INITU)
(check-expect (add-new-world A-UNIV iworld3)    ADD-AUNIV)
(check-expect (add-new-world A-UNIV iworld1)    ADD-REPEAT)

;; Tests using sample values for add-new-world
(check-expect (add-new-world (list iworld1) iworld2)
              (make-bundle (list iworld2 iworld1)
                           (list (make-mail iworld1 "iworld2 has joined"))
                           '()))
```

Intro to Distributed Programming

Server

- ```
  ;; universe iworld → bundle
  ;; Purpose: Add the given world to the universe
  ;; ASSUMPTION: The name of the new iworld is a string
  (define (add-new-world a-universe an-iworld)
   (if (member? (iworld-name an-iworld) (map iworld-name a-universe))
       a-universe
       (local
        [(define new-univ (cons an-iworld a-universe))
         (define new-mails
                 (map
                  (λ (iw)
                   (make-mail
                    iw
                    (string-append (iworld-name an-iworld)
                                   " has joined")))
                  a-universe))]
        (make-bundle new-univ new-mails '()))))
  ```
  ```

Intro to Distributed Programming

Server

- The incoming arrows to the server in the protocol diagrams inform us that the server only receives messages containing `tms`
- The handler for processing messages only needs to forward an incoming message to all the worlds except the sender
- The `universe` must be filtered to exclude the `iworld` that sent the message and `mails` must be created for the remaining `iworlds`

Intro to Distributed Programming

Server

```
;; Sample Expressions for process-message
(define AUNIV-MESS
  (local [(define new-mails
            (map (λ (iw) (make-mail iw "Hi!"))
                 (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                              (iworld-name iw))))
                         A-UNIV)))]
    (make-bundle A-UNIV new-mails '())))

(define IUNIV-MESS
  (local [(define new-mails
            (map (λ (iw) (make-mail iw "Hi!"))
                 (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                              (iworld-name iw))))
                         INIT-UNIV)))]
    (make-bundle INIT-UNIV new-mails '())))
```

Intro to Distributed Programming

Server

- ```
  ;; universe iworld message → bundle
  ;; Purpose: To process the given message from the given world
  (define (process-message a-univ an-iw a-mess)
  ```

- ```
  ;; Sample Expressions for process-message
  (define AUNIV-MESS
    (local [(define new-mails
              (map (λ (iw) (make-mail iw "Hi!"))
                   (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                                (iworld-name iw))))
                           A-UNIV)))]
      (make-bundle A-UNIV new-mails '())))

  (define IUNIV-MESS
    (local [(define new-mails
              (map (λ (iw) (make-mail iw "Hi!"))
                   (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                                (iworld-name iw))))
                           INIT-UNIV)))]
      (make-bundle INIT-UNIV new-mails '())))
  ```

Intro to Distributed Programming

Server

- ```
  ;; universe iworld message → bundle
  ;; Purpose: To process the given message from the given world
  (define (process-message a-univ an-iw a-mess)
  ```

- ```
  ;; Sample Expressions for process-message
  (define AUNIV-MESS
    (local [(define new-mails
              (map (λ (iw) (make-mail iw "Hi!"))
                   (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                                (iworld-name iw))))
                           A-UNIV)))]
      (make-bundle A-UNIV new-mails '())))

  (define IUNIV-MESS
    (local [(define new-mails
              (map (λ (iw) (make-mail iw "Hi!"))
                   (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                                (iworld-name iw))))
                           INIT-UNIV)))]
      (make-bundle INIT-UNIV new-mails '())))
  ```

- ```
  ;; Tests using sample computations for process-message
  (check-expect (process-message A-UNIV iworld2 "Hi!")    AUNIV-MESS)
  (check-expect (process-message INIT-UNIV iworld2 "Hi!") IUNIV-MESS)
  ;; Tests using sample values for process-message
  (check-expect (process-message (list iworld2 iworld3) iworld3 "OK")
                (make-bundle (list iworld2 iworld3) (list (make-mail iworld2 "OK")) '()))
  ```
  ```

Intro to Distributed Programming

Server

```
;; universe iworld message → bundle
;; Purpose: To process the given message from the given world
(define (process-message a-univ an-iw a-mess)
  (local [(define new-mails (map (λ (iw) (make-mail iw a-mess))
                                 (filter
                                   (λ (iw)
                                     (not (equal? (iworld-name an-iw)
                                                  (iworld-name iw))))
                                   a-univ)))]
    (make-bundle a-univ new-mails '())))
```

```
;; Sample Expressions for process-message
(define AUNIV-MESS
  (local [(define new-mails
            (map (λ (iw) (make-mail iw "Hi!"))
                 (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                              (iworld-name iw))))
                         A-UNIV)))]
    (make-bundle A-UNIV new-mails '())))

(define IUNIV-MESS
  (local [(define new-mails
            (map (λ (iw) (make-mail iw "Hi!"))
                 (filter (λ (iw) (not (equal? (iworld-name iworld2)
                                              (iworld-name iw))))
                         INIT-UNIV)))]
    (make-bundle INIT-UNIV new-mails '())))
```

```
;; Tests using sample computations for process-message
(check-expect (process-message A-UNIV iworld2 "Hi!")    AUNIV-MESS)
(check-expect (process-message INIT-UNIV iworld2 "Hi!") IUNIV-MESS)
;; Tests using sample values for process-message
(check-expect (process-message (list iworld2 iworld3) iworld3 "OK")
              (make-bundle (list iworld2 iworld3) (list (make-mail iworld2 "OK")) '()))
```

Intro to Distributed Programming

Server

- To run the chat tool on your machine first run the server and then run one or more clients
- Type messages in each of the clients and see how the messages sent appear in the other clients

Intro to Distributed Programming

Server

- To run the chat tool on your machine first run the server and then run one or more clients
- Type messages in each of the clients and see how the messages sent appear in the other clients
- Once you are fairly sure that the chat tool is working you may now use it to chat with your fellow classmates
- Pick a classmate to run the server and get their internet address
- All clients need to substitute `LOCALHOST` with a string containing the internet address of the classmate running the server

Intro to Distributed Programming

Homework

- Problems: 326–328

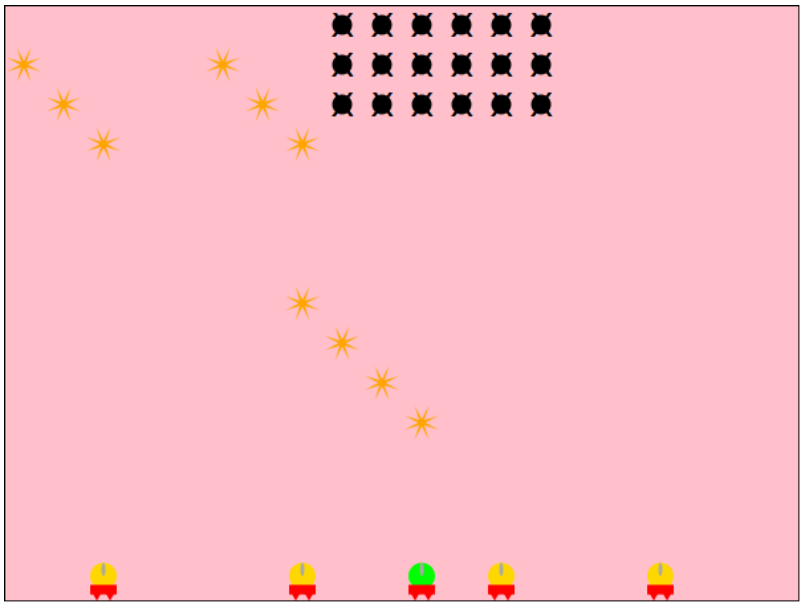
Aliens Attack Version 6

- Our next goal is to refine Aliens Attack to allow for multiple players.
- Will explore how to design a distributed program using both a thin and a thick server

Aliens Attack Version 6

- Our next goal is to refine Aliens Attack to allow for multiple players.
- Will explore how to design a distributed program using both a thin and a thick server
- First a single-player game refinement that makes incorporating multiple players easier
- The goal is to have all the `world`-related data definitions needed for multiple players in place and implemented for a single-player game

Aliens Attack Version 6



- Our next goal is to refine Aliens Attack to allow for multiple players.
- Will explore how to design a distributed program using both a thin and a thick server
- First a single-player game refinement that makes incorporating multiple players easier
- The goal is to have all the `world`-related data definitions needed for multiple players in place and implemented for a single-player game
- Consider the snapshot of multiplayer Aliens Attack
- The game has multiple `rockets`
- The rest of the elements remain the same
- No need for a server, a communication protocol, and all the other necessary features for a distributed program

Aliens Attack Version 6

Refining the `world` Data Definition

- Need to have multiple rockets that are all allies
- Tempting to simply change the `world` data definition to have allies instead of a single rocket
- We need, however, a more detailed problem analysis
- Think about how a player ought to start the game
- Should a player start, as in the single player game, with a world that has a full army of aliens moving in some hardwired direction with no shots?

Aliens Attack Version 6

Refining the `world` Data Definition

- Need to have multiple rockets that are all allies
- Tempting to simply change the `world` data definition to have allies instead of a single rocket
- We need, however, a more detailed problem analysis
- Think about how a player ought to start the game
- Should a player start, as in the single player game, with a world that has a full army of aliens moving in some hardwired direction with no shots?
- After the first, a player cannot start with a world that has a full army of aliens and no shots because the first player may have already started shooting and neutralizing invaders
- This means that the starting `world` for a player must be provided by the server

Aliens Attack Version 6

Refining the `world` Data Definition

- There must be variety in the `world` data definition
- Player joins: `world` is uninitialized and server provides value

Aliens Attack Version 6

Refining the `world` Data Definition

- There must be variety in the `world` data definition
- Player joins: `world` is uninitialized and server provides value
- ```
  ;; A world is either
  ;;  1. 'uninitialized
  ;;  2. a structure: (make-world lor loa dir los)
  (define-struct world (allies aliens dir shots))
  ```
  ```

Aliens Attack Version 6

Refining the world Data Definition

- There must be variety in the world data definition
- Player joins: world is uninitialized and server provides value
- ```
;; A world is either
;; 1. 'uninitialized
;; 2. a structure: (make-world lor loa dir los)
(define-struct world (allies aliens dir shots))
```
- ```
;; world ... → ...   Purpose:
;; (define (f-on-world w ...)
;;   (if (eq? a-world 'uninitialized)
;;       ...
;;       (... (world-allies w)... (world-aliens w)
;;        ... (world-dir w)... (world-shots w))))
;;  Sample instances of world
;;  (define WORLD1  'uninitialized)
;;  (define WORLD2  (make-world ... ... ... ...))
;;  Sample expressions for f-on-world
;;  (define WORLD1-VAL ... WORLD1 ...)
;;  (define WORLD2-VAL ... WORLD2 ...) ...
;;  ;; Tests using sample computations for f-on-world
;;  (check-expect (f-on-world WORLD1 ...) WORLD1-VAL)
;;  (check-expect (f-on-world WORLD2 ...) WORLD2-VAL) ...
;;  Tests using sample values for f-on-world
;;  (check-expect (f-on-world ... ...) ... ) ...
```

Aliens Attack Version 6

Refining the `world` Data Definition

- What is a `lor`?

Aliens Attack Version 6

Refining the `world` Data Definition

- What is a `lor`?
- ```
  ;; An lor is a (listof ally)
  ```
- What is an `ally`?
  ```

Aliens Attack Version 6

Refining the `world` Data Definition

- What is a `lor`?
- ```
  ;; An lor is a (listof ally)
  ```
- What is an `ally`?
- Need to distinguish which player owns each rocket
  ```

Aliens Attack Version 6

Refining the `world` Data Definition

- What is a `lor`?
- `;; An lor is a (listof ally)`
- What is an `ally`?
- Need to distinguish which player owns each rocket
- ```
  ;; An ally is a structure, (make-ally rocket string), with
  ;; a player's rocket and name
  (define-struct ally (rocket name))
  ```
  ```

Aliens Attack Version 6

Refining the world Data Definition

- What is a lor?
- ```
  ;; An lor is a (listof ally)
  ```
- What is an ally?
- Need to distinguish which player owns each rocket
- ```
  ;; An ally is a structure, (make-ally rocket string), with
  ;; a player's rocket and name
  (define-struct ally (rocket name))
  ```
- ```
  ;; ally ... → ...   Purpose:
  ;; (define (f-on-ally an-ally ...)
  ;;   (... (world-allies an-ally) ... (world-aliens an-ally)
  ;;    ... (world-dir an-ally) ... (world-shots an-ally))))
  ;;
  ;;  Sample instances of ally
  ;;  (define ALLY1  (make-ally ... ...))
  ;;
  ;;  Sample expressions for f-on-ally
  ;;  (define ALLY1-VAL ... ALLY1 ...) ...
  ;;
  ;;  Tests using sample computations for f-on-ally
  ;;  (check-expect (f-on-ally ALLY1 ...) ALLY-VAL) ...
  ;;
  ;;  Tests using sample values for f-on-ally
  ;;  (check-expect (f-on-ally ... ...) ...) ...
  ```
  ```

Aliens Attack Version 6

Refining the `world` Data Definition

- Sample instances:

```
(define MY-NAME "Yoli Ortega")

(define INIT-ALLY  (make-ally INIT-ROCKET  MY-NAME))
(define INIT-ALLY2 (make-ally INIT-ROCKET2 MY-NAME))

(define INIT-ALLIES  (list INIT-ALLY))
(define INIT-ALLIES2 (list INIT-ALLY2))

(define INIT-WORLD  (make-world INIT-ALLIES INIT-LOA INIT-DIR INIT-LOS))
(define INIT-WORLD2 (make-world INIT-ALLIES2 (list INIT-ALIEN2) DIR2 (list SHOT2)))
(define WORLD3 (make-world (list (make-ally 7 MY-NAME))
                           (list (make-posn 3 3))
                           'right
                           (list (make-posn 3 3))))
(define UNINIT-WORLD 'uninitialized)
```

Aliens Attack Version 6

Refining the world Data Definition

- Sample instances:

```
(define MY-NAME "Yoli Ortega")

(define INIT-ALLY  (make-ally INIT-ROCKET  MY-NAME))
(define INIT-ALLY2 (make-ally INIT-ROCKET2 MY-NAME))

(define INIT-ALLIES  (list INIT-ALLY))
(define INIT-ALLIES2 (list INIT-ALLY2))

(define INIT-WORLD  (make-world INIT-ALLIES INIT-LOA INIT-DIR INIT-LOS))
(define INIT-WORLD2 (make-world INIT-ALLIES2 (list INIT-ALIEN2) DIR2 (list SHOT2)))
(define WORLD3 (make-world (list (make-ally 7 MY-NAME))
                           (list (make-posn 3 3))
                           'right
                           (list (make-posn 3 3))))
(define UNINIT-WORLD 'uninitialized)
```

- The run function:

```
;; string → world   Purpose: To run the game
(define (run a-name)   dummy parameter
  (local [(define TICK-RATE 1/4)]
    (big-bang
      INIT-WORLD  no server in this version
      [on-draw draw-world]
      [name MY-NAME]
      [on-key process-key]
      [on-tick process-tick TICK-RATE]
      [stop-when game-over? draw-last-world])))
```

Aliens Attack Version 6

The `draw-world` Refinement

- ```
  ;; Sample expressions for draw-world
  (define WORLD-SCN1  (if (eq? INIT-WORLD UNINIT-WORLD)
                          E-SCENE
                          (draw-world INIT-WORLD)))

  (define WORLD-SCN2  (if (eq? UNINIT-WORLD UNINIT-WORLD)
                          E-SCENE
                          (draw-world UNINIT-WORLD)))
  ```
  ```

Aliens Attack Version 6

The draw-world Refinement

- Basic local changes:

```
(define E-SCENE-COLOR 'pink)
(define E-SCENE (empty-scene E-SCENE-W E-SCENE-H E-SCENE-COLOR))
;; world → scene   Purpose: To draw the world in E-SCENE
(define (draw-world a-world)
 (local
  [(define FUSELAGE-COLOR2 'gold)
   (define FUSELAGE2 (mk-fuselage-img FUSELAGE-COLOR2))
   (define ROCKET-MAIN2 (mk-rocket-main-img WINDOW
                                           FUSELAGE2
                                           BOOSTER))
   (define ROCKET-IMG2 (mk-rocket-ci ROCKET-MAIN2 NACELLE))
   ...]
  (if (eq? a-world UNINIT-WORLD)
      E-SCENE
      (draw-world a-world))))
```

Aliens Attack Version 6

The draw-world Refinement

```
;; world → scene   Purpose: To draw the world in E-SCENE
(define (draw-world a-world)
  (local
   [;; world → scene
    ;; Purpose: To draw the world in E-SCENE
    ;; ASSUMPTION: The given world is a structure
    (define (draw-world a-world)
     (draw-los (world-shots a-world)
               (draw-loa (world-aliens a-world)
                         (draw-allies (world-allies a-world)
                                      E-SCENE))))
          ...]
   (if (eq? a-world UNINIT-WORLD)
       E-SCENE
       (draw-world a-world))))
```

Aliens Attack Version 6

The draw-world Refinement

```
;; world → scene   Purpose: To draw the world in E-SCENE
(define (draw-world a-world)
  (local [;;lor scene → scene Purpose: Draw allies in given scene
          (define draw-allies
            (draw-lox-maker
             (λ (an-ally scn)
              (if (string=? (ally-name an-ally) MY-NAME)
                  (draw-rocket (ally-rocket an-ally) scn)
                  (draw-ally   (ally-rocket an-ally) scn)))))
          ...]
    (if (eq? a-world UNINIT-WORLD)
        E-SCENE
        (draw-world a-world))))
```

Aliens Attack Version 6

The draw-world Refinement

```
;; world → scene   Purpose: To draw the world in E-SCENE
(define (draw-world a-world)
 (local
  [;; image → (rocket scene → scene)
   ;; Purpose: Create a rocket drawing function
   (define (draw-ally-maker rocket-img)
    (local [;; rocket scene → scene
            ;; Purpose: To draw the rocket in the given scene
            (define (draw-ally a-rocket a-scene)
              (draw-ci rocket-img a-rocket ROCKET-Y a-scene))]
      draw-ally))

   ;; rocket scene → scene
   ;; Purpose: Draw the rocket in given scene
   (define draw-rocket (draw-ally-maker ROCKET-IMG))

   ;; rocket scene → scene   Purpose: Draw rocket in given scene
   (define draw-ally (draw-ally-maker ROCKET-IMG2))
   ...]
   (if (eq? a-world UNINIT-WORLD)
       E-SCENE
       (draw-world a-world))))
```

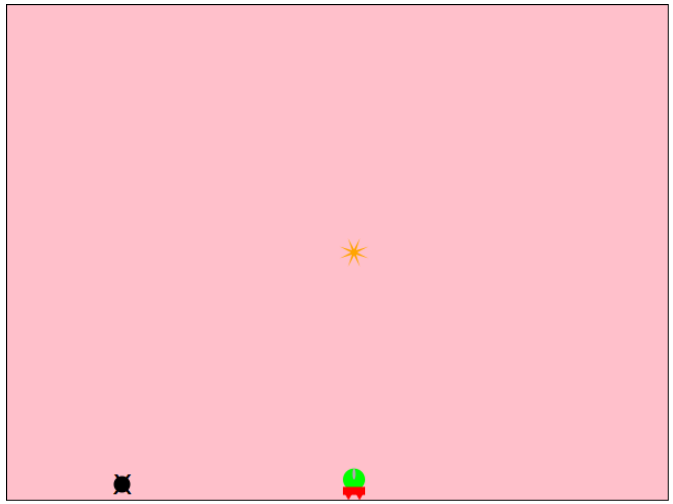
Aliens Attack Version 6

The `draw-world` Refinement

- The final step is to refine the tests for `draw-world` as follows:

```
;; Tests using sample computations for draw-world
(check-expect (draw-world INIT-WORLD)   WORLD-SCN1)
(check-expect (draw-world UNINIT-WORLD) WORLD-SCN2)

;; Tests using sample computations for draw-world
;; Added NO-SHOT to fully test draw-shot
(check-expect (draw-world (make-world (list INIT-ALLY)
                                      (list INIT-ALIEN2)
                                      DIR2
                                      (list SHOT2 NO-SHOT)))
```



)

Aliens Attack Version 6

The process-key Refinement

- The process-key handler from Aliens Attack version 5:

```
;; world key → world  Purpose: Process a key event to return next world
(define (process-key a-world a-key)
  (local [...]
    (process-key a-world a-key)))
```

Aliens Attack Version 6

The process-key Refinement

- The process-key handler from Aliens Attack version 5:

```
;; world key → world  Purpose: Process a key event to return next world
(define (process-key a-world a-key)
  (local [...]
    (process-key a-world a-key)))
```

- A conditional expression is now needed a world:

```
;; Sample expressions for process-key
(define KEY-RVAL (if (eq? INIT-WORLD UNINIT-WORLD)
                     INIT-WORLD
                     (process-key INIT-WORLD "right")))
(define KEY-LVAL (if (eq? INIT-WORLD UNINIT-WORLD)
                     INIT-WORLD
                     (process-key INIT-WORLD "left")))
(define KEY-SVAL (if (eq? INIT-WORLD UNINIT-WORLD)
                     INIT-WORLD
                     (process-key INIT-WORLD " ")))
(define KEY-SVAL2 (if (eq? INIT-WORLD2 UNINIT-WORLD)
                      INIT-WORLD2
                      (process-key INIT-WORLD2 " ")))
(define KEY-OVAL (if (eq? INIT-WORLD2 UNINIT-WORLD)
                     INIT-WORLD2
                     (process-key INIT-WORLD2 "m")))
(define KEY-NAOA (if (eq? UNINIT-WORLD UNINIT-WORLD)
                     UNINIT-WORLD
                     (process-key UNINIT-WORLD " ")))
```

Aliens Attack Version 6

The process-key Refinement

- The process-key handler from Aliens Attack version 5:

```
;; world key → world  Purpose: Process a key event to return next world
(define (process-key a-world a-key)
  (local [...]
    (process-key a-world a-key)))
```

- A conditional expression is now needed a world:

```
;; Sample expressions for process-key
(define KEY-RVAL (if (eq? INIT-WORLD UNINIT-WORLD)
                     INIT-WORLD
                     (process-key INIT-WORLD "right")))
(define KEY-LVAL (if (eq? INIT-WORLD UNINIT-WORLD)
                     INIT-WORLD
                     (process-key INIT-WORLD "left")))
(define KEY-SVAL (if (eq? INIT-WORLD UNINIT-WORLD)
                     INIT-WORLD
                     (process-key INIT-WORLD " ")))
(define KEY-SVAL2 (if (eq? INIT-WORLD2 UNINIT-WORLD)
                      INIT-WORLD2
                      (process-key INIT-WORLD2 " ")))
(define KEY-OVAL (if (eq? INIT-WORLD2 UNINIT-WORLD)
                     INIT-WORLD2
                     (process-key INIT-WORLD2 "m")))
(define KEY-NAOA (if (eq? UNINIT-WORLD UNINIT-WORLD)
                     UNINIT-WORLD
                     (process-key UNINIT-WORLD " ")))
```

- Abstraction over the sample expressions yields a handler with the following structure:

```
;; world key → world  Purpose: Process a key event to return next world
(define (process-key a-world a-key)
  (local [...]
    (if (eq? a-world UNINIT-WORLD)
        a-world
        (process-key a-world a-key))))
```

Aliens Attack Version 6

The `process-key` Refinement

- The tests using sample computations do not need to be updated

Aliens Attack Version 6

The process-key Refinement

- The tests using sample computations do not need to be updated
- The tests using sample values need to properly construct worlds
-
```
;; Tests using sample values for process-key
(check-expect
  (process-key (make-world (list (make-ally (sub1 MAX-CHARS-HORIZONTAL) MY-NAME))
                                 INIT-LOA
                                 'right
                                 INIT-LOS)
               "right")
  (make-world (list (make-ally (sub1 MAX-CHARS-HORIZONTAL) MY-NAME))
                    INIT-LOA
                    'right
                    INIT-LOS))
(check-expect
  (process-key (make-world (list (make-ally 0 MY-NAME)) INIT-LOA 'left INIT-LOS)
               "left")
  (make-world (list (make-ally 0 MY-NAME)) INIT-LOA 'left INIT-LOS))
(check-expect
  (process-key (make-world (list (make-ally 0 MY-NAME)) INIT-LOA 'left INIT-LOS)
                           "o")
  (make-world (list (make-ally 0 MY-NAME)) INIT-LOA 'left INIT-LOS))
(check-expect (process-key INIT-WORLD2 ";") INIT-WORLD2)
(check-expect
 (process-key (make-world (list (make-ally 0 MY-NAME)) INIT-LOA 'left INIT-LOS)
              " ")
 (make-world (list(make-ally 0 MY-NAME))INIT-LOA 'left (cons (make-posn 0 MAX-IMG-Y) '())
(check-expect
  (process-key (make-world (list (make-ally 0 MY-NAME)) INIT-LOA 'left (cons SHOT2 '()))
               "left")
  (make-world (list (make-ally 0 MY-NAME)) INIT-LOA 'left (cons SHOT2 '())))
```

Aliens Attack Version 6

The `process-key` Refinement

- Refine local function `process-key` that processes a `world`
- Player moves her rocket then the right ally is moved
- Player shoots then right rocket's `image-x` is needed

Aliens Attack Version 6

The `process-key` Refinement

- Refine local function `process-key` that processes a `world`
- Player moves her rocket then the right ally is moved
- Player shoots then right rocket's `image-x` is needed
- ```
  ;; world key → world  Purpose: Process a key event to return next world
  ;; ASSUMPTION: The given world is a structure
  (define (process-key a-world a-key)
  ```
  ```

Aliens Attack Version 6

The process-key Refinement

- Refine local function process-key that processes a world
- Player moves her rocket then the right ally is moved
- Player shoots then right rocket's image-x is needed
- ```
  ;; world key → world  Purpose: Process a key event to return next world
  ;; ASSUMPTION: The given world is a structure
  (define (process-key a-world a-key)
  ```
- ```
   (cond [(key=? a-key "right")
          (make-world (move-ally-right MY-NAME
                                       (world-allies a-world))
                      (world-aliens a-world)
                      (world-dir a-world)
                      (world-shots a-world))]
  ```

Aliens Attack Version 6

The process-key Refinement

- Refine local function process-key that processes a world
- Player moves her rocket then the right ally is moved
- Player shoots then right rocket's image-x is needed
- ```
  ;; world key → world  Purpose: Process a key event to return next world
  ;; ASSUMPTION: The given world is a structure
  (define (process-key a-world a-key)
  ```
- ```
   (cond [(key=? a-key "right")
          (make-world (move-ally-right MY-NAME
                                       (world-allies a-world))
                      (world-aliens a-world)
                      (world-dir a-world)
                      (world-shots a-world))]
  ```
- ```
          [(key=? a-key "left")
           (make-world (move-ally-left  MY-NAME
                                        (world-allies a-world))
                       (world-aliens a-world)
                       (world-dir a-world)
                       (world-shots a-world))]
  ```
  ```

Aliens Attack Version 6

The process-key Refinement

- Refine local function process-key that processes a world
- Player moves her rocket then the right ally is moved
- Player shoots then right rocket's image-x is needed
- ```
  ;; world key → world  Purpose: Process a key event to return next world
  ;; ASSUMPTION: The given world is a structure
  (define (process-key a-world a-key)
  ```
- ```
   (cond [(key=? a-key "right")
          (make-world (move-ally-right MY-NAME
                                       (world-allies a-world))
                      (world-aliens a-world)
                      (world-dir a-world)
                      (world-shots a-world))]
  ```
- ```
          [(key=? a-key "left")
           (make-world (move-ally-left  MY-NAME
                                        (world-allies a-world))
                       (world-aliens a-world)
                       (world-dir a-world)
                       (world-shots a-world))]
  ```
- ```
          [(key=? a-key " ")
           (make-world (world-allies a-world)
                       (world-aliens a-world)
                       (world-dir a-world)
                       (cons (process-shooting
                              (ally-rocket
                               (get-ally MY-NAME
                                         (world-allies a-world))))
                             (world-shots a-world)))]
  ```

Aliens Attack Version 6

The process-key Refinement

- Refine local function process-key that processes a world
- Player moves her rocket then the right ally is moved
- Player shoots then right rocket's image-x is needed
- ```
  ;; world key → world  Purpose: Process a key event to return next world
  ;; ASSUMPTION: The given world is a structure
  (define (process-key a-world a-key)
  ```
- ```
   (cond [(key=? a-key "right")
          (make-world (move-ally-right MY-NAME
                                       (world-allies a-world))
                      (world-aliens a-world)
                      (world-dir a-world)
                      (world-shots a-world))]
  ```
- ```
          [(key=? a-key "left")
           (make-world (move-ally-left  MY-NAME
                                        (world-allies a-world))
                       (world-aliens a-world)
                       (world-dir a-world)
                       (world-shots a-world))]
  ```
- ```
          [(key=? a-key " ")
           (make-world (world-allies a-world)
                       (world-aliens a-world)
                       (world-dir a-world)
                       (cons (process-shooting
                              (ally-rocket
                               (get-ally MY-NAME
                                         (world-allies a-world))))
                             (world-shots a-world)))]
  ```
- ```
          [else a-world]))
  ```
- Three new auxiliary functions are needed: move-ally-right, move-ally-left, and get-ally.
  ```

Aliens Attack Version 6

The `process-key` Refinement

- `get-ally`: extract the ally that has `MY-NAME`

Aliens Attack Version 6

The `process-key` Refinement

- `get-ally`: extract the ally that has `MY-NAME`
- ```
  ;; string lor → ally
  ;; Purpose: Extract ally with given name
  ;; ASSUMPTIONS: There is a single ally with given name
  (define (get-ally a-name a-lor)
    (first (filter (λ (an-ally)
                     (string=? a-name (ally-name an-ally)))
                   a-lor)))
  ```
- Assumed that there is a single ally with the given name
- Means that the list returned by `filter` contains a single ally
- Extract the ally using `first`
  ```

Aliens Attack Version 6

The `process-key` Refinement

- The functions to move the player's rocket are virtually the same
- Only vary by moving function: `move-rckt-right` or `move-rckt-left`
- Suggests creating a curried function for specialized `ally`-moving functions

Aliens Attack Version 6

The process-key Refinement

- The functions to move the player's rocket are virtually the same
- Only vary by moving function: `move-rckt-right` or `move-rckt-left`
- Suggests creating a curried function for specialized `ally`-moving functions
- ```
  ;; (rocket → rocket) → (string lor → lor)
  ;; Purpose: Make an ally-moving function using given function
  (define (make-ally-mover move-rckt)
    ;; string lor → (ally → lor)
    ;; Purpose: Move ally with given name using rocket-moving funct
    (λ (a-name a-lor)
      (map (λ (an-ally)
             (if (string=? a-name (ally-name an-ally))
                 (make-ally (move-rckt (ally-rocket an-ally))
                            (ally-name an-ally))
                 an-ally))
           a-lor)))
  ```
  ```

Aliens Attack Version 6

The process-key Refinement

- The functions to move the player's rocket are virtually the same
- Only vary by moving function: `move-rckt-right` or `move-rckt-left`
- Suggests creating a curried function for specialized `ally`-moving functions
- ```
  ;; (rocket → rocket) → (string lor → lor)
  ;; Purpose: Make an ally-moving function using given function
  (define (make-ally-mover move-rckt)
    ;; string lor → (ally → lor)
    ;; Purpose: Move ally with given name using rocket-moving funct
    (λ (a-name a-lor)
      (map (λ (an-ally)
             (if (string=? a-name (ally-name an-ally))
                 (make-ally (move-rckt (ally-rocket an-ally))
                            (ally-name an-ally))
                 an-ally))
           a-lor)))
  ```
- The `ally`-moving functions are implemented as follows:

  ```
  ;; string lor → lor   Purpose: Move ally with given name right
  (define move-ally-right (make-ally-mover move-rckt-right))

  ;; string lor → lor   Purpose: Move ally with given name left
  (define move-ally-left (make-ally-mover move-rckt-left))
  ```
  ```

Aliens Attack Version 6

The `process-tick` Refinement

- `process-tick` must be refined

Aliens Attack Version 6

The process-tick Refinement

- process-tick must be refined
- The sample expressions may be refined to be:

```
;; Sample expressions for process-tick
(define AFTER-TICK-WORLD1  (if (eq? INIT-WORLD UNINIT-WORLD)
                               INIT-WORLD
                               (process-tick INIT-WORLD)))

(define AFTER-TICK-WORLD2  (if (eq? INIT-WORLD2 UNINIT-WORLD)
                               INIT-WORLD2
                               (process-tick INIT-WORLD2)))

(define AFTER-TICK-UNINITW  (if (eq? UNINIT-WORLD UNINIT-WORLD)
                                UNINIT-WORLD
                                (process-tick UNINIT-WORLD)))
```

Aliens Attack Version 6

The process-tick Refinement

- process-tick must be refined
- The sample expressions may be refined to be:

```
;; Sample expressions for process-tick
(define AFTER-TICK-WORLD1  (if (eq? INIT-WORLD UNINIT-WORLD)
                               INIT-WORLD
                               (process-tick INIT-WORLD)))

(define AFTER-TICK-WORLD2  (if (eq? INIT-WORLD2 UNINIT-WORLD)
                               INIT-WORLD2
                               (process-tick INIT-WORLD2)))

(define AFTER-TICK-UNINITW  (if (eq? UNINIT-WORLD UNINIT-WORLD)
                                UNINIT-WORLD
                                (process-tick UNINIT-WORLD)))
```

- Abstracting over the sample expression yields a new process-tick with the following structure:

```
;; world → world
;; Purpose: Create a new world after a clock tick
(define (process-tick a-world)
  (local [...]
    (if (eq? a-world UNINIT-WORLD)
        a-world
        (process-tick a-world))))
```

Aliens Attack Version 6

The process-tick Refinement

- ```
  (check-expect
   (process-tick
    (make-world
     (list (make-ally INIT-ROCKET MY-NAME)) (cons (make-posn 1 5) '()) 'left INIT-LOS))
    (make-world
     (list (make-ally INIT-ROCKET MY-NAME)) (cons (make-posn 0 5) '()) 'down INIT-LOS))
  (check-expect (process-tick (make-world
                               (list (make-ally INIT-ROCKET MY-NAME))
                               (list (make-posn 2 5))
                               'left
                               (list (make-posn 1 6) NO-SHOT)))
                (make-world   (list (make-ally INIT-ROCKET MY-NAME)) '() 'left '()))
  (check-expect (process-tick (make-world
                               (list (make-ally INIT-ROCKET2 MY-NAME))
                               (list (make-posn (- MAX-CHARS-HORIZONTAL 2) 10))
                               'right
                               (list SHOT2)))
                (make-world (list (make-ally INIT-ROCKET2 MY-NAME))
                            (cons (make-posn MAX-IMG-X 10) '())
                            'down
                            (list (make-posn (posn-x SHOT2) (sub1 (posn-y SHOT2))))))
  (check-expect
   (process-tick (make-world (list (make-ally INIT-ROCKET2 MY-NAME))
                             (list (make-posn MAX-IMG-X 2))
                             'down
                             (list (make-posn 15 6))))
                 (make-world (list (make-ally INIT-ROCKET2 MY-NAME))
                             (list (make-posn MAX-IMG-X 3))
                             'left
                             (list (make-posn 15 5))))
  ```
  ```

Aliens Attack Version 6

The `process-tick` Refinement

- The local `process-tick` function must also be updated
- It's refinement only needs to use the proper selector for the allies

Aliens Attack Version 6

The process-tick Refinement

- The local process-tick function must also be updated
- It's refinement only needs to use the proper selector for the allies
- ```
  ;; world → world
  ;; Purpose: Create a new world after a clock tick
  ;; ASSUMPTION: The given world is a structure
  (define (process-tick a-world)
    (make-world
      (world-allies a-world)
      (remove-hit-aliens (move-loa (world-aliens a-world)
                                   (world-dir a-world))
                         (move-los (world-shots a-world)))
      (new-dir-after-tick (move-loa (world-aliens a-world)
                                    (world-dir a-world))
                          (world-dir a-world))
      (remove-shots (move-los (world-shots a-world))
                    (move-loa (world-aliens a-world)
                              (world-dir a-world)))))
  ```
- Once again the assumption that the given world is a structure is made explicit for the benefit of any reader of the code
  ```

Aliens Attack Version 6

The `game-over?` Refinement

- The `game-over?` returns `#false`, as before, when the given `world`'s list of `aliens` is empty or when any of the `aliens` has reached earth
- `t` must return `#false` if the given `world` is uninitialized

Aliens Attack Version 6

The game-over? Refinement

- Updated sample expressions:

```
(define GAME-OVER1
  (and (not (eq? INIT-WORLD2 UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens INIT-WORLD2))
       (empty? (world-aliens INIT-WORLD2)))))

(define GAME-OVER2
  (and (not (eq? WORLD3 UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens WORLD3))
       (empty? (world-aliens WORLD3)))))

(define GAME-NOT-OVER
  (and (not (eq? INIT-WORLD UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens INIT-WORLD))
       (empty? (world-aliens INIT-WORLD)))))

(define GAME-NOT-DONE
  (and (not (eq? UNINIT-WORLD UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens UNINIT-WORLD))
       (empty? (world-aliens UNINIT-WORLD)))))
```

Aliens Attack Version 6

The game-over? Refinement

- Updated sample expressions:

```
(define GAME-OVER1
  (and (not (eq? INIT-WORLD2 UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens INIT-WORLD2))
       (empty? (world-aliens INIT-WORLD2)))))

(define GAME-OVER2
  (and (not (eq? WORLD3 UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens WORLD3))
       (empty? (world-aliens WORLD3)))))

(define GAME-NOT-OVER
  (and (not (eq? INIT-WORLD UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens INIT-WORLD))
       (empty? (world-aliens INIT-WORLD)))))

(define GAME-NOT-DONE
  (and (not (eq? UNINIT-WORLD UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens UNINIT-WORLD))
       (empty? (world-aliens UNINIT-WORLD)))))
```

- Abstracting over the sample expressions yields the refined handler:

```
;; world → Boolean
;; Purpose: Detect if the game is over
(define (game-over? a-world)
  (and (not (eq? a-world UNINIT-WORLD))
       (or (ormap (λ (an-alien) (= (posn-y an-alien) MAX-IMG-Y))
                  (world-aliens a-world))
       (empty? (world-aliens a-world)))))
```

Aliens Attack Version 6

The game-over? Refinement

- Tests using sample values must be updated to construct worlds that have a lor:

```
;; Tests using sample values for game-over?
(check-expect (game-over?
                (make-world (list (make-ally 8 MY-NAME))
                            (list (make-posn 0 3))
                            'right
                            NO-SHOT))
              #false)
(check-expect (game-over?
                (make-world (list (make-ally 8 MY-NAME))
                            (list (make-posn 0 MAX-IMG-Y))
                            'right
                            (list (make-posn 12 11))))
              #true)
(check-expect (game-over?
                (make-world (list (make-ally 8 MY-NAME))
                            (list (make-posn 0 5))
                            'right
                            (list (make-posn 0 5))))
              #false)
```

Aliens Attack Version 6

Homework

- Problems: 329–330

Aliens Attack Version 7

- Let's refine Aliens Attack 6 into a multiplayer game
- The design recipe for distributed programming helps you manage the complexity of its development
- Take time to understand how the different steps are interrelated

Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs

Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs
- A message is sent to the server when the player shoots or moves the rocket
- A message is received from the server when it joins the game and when another player shoots, moves it rocket, arrives, or departs

Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs
- A message is sent to the server when the player shoots or moves the rocket
- A message is received from the server when it joins the game and when another player shoots, moves it rocket, arrives, or departs
- The server receives messages from a player, manages the joining of new players, and manages the departure of players

Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs
- A message is sent to the server when the player shoots or moves the rocket
- A message is received from the server when it joins the game and when another player shoots, moves it rocket, arrives, or departs
- The server receives messages from a player, manages the joining of new players, and manages the departure of players
- When message arrives from a player, the server broadcasts the message to the other players

Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs
- A message is sent to the server when the player shoots or moves the rocket
- A message is received from the server when it joins the game and when another player shoots, moves it rocket, arrives, or departs
- The server receives messages from a player, manages the joining of new players, and manages the departure of players
- When message arrives from a player, the server broadcasts the message to the other players
- When a player joins the game the server provides it with its starting world and broadcasts a message to all players that they have a new `ally`
- The starting world depends on when $\texttt{Player}_i$ arrives
- $\texttt{Player}_i$ is the first player: sends the initial `world`
- Otherwise: server requests `world` existing player and sends it to $\texttt{Player}_i$

Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs
- A message is sent to the server when the player shoots or moves the rocket
- A message is received from the server when it joins the game and when another player shoots, moves it rocket, arrives, or departs
- The server receives messages from a player, manages the joining of new players, and manages the departure of players
- When message arrives from a player, the server broadcasts the message to the other players
- When a player joins the game the server provides it with its starting world and broadcasts a message to all players that they have a new `ally`
- The starting world depends on when $\texttt{Player}_i$ arrives
- $\texttt{Player}_i$ is the first player: sends the initial `world`
- Otherwise: server requests `world` existing player and sends it to $\texttt{Player}_i$
- In order to properly select recipients of messages each player component must have a distinct name

Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs
- A message is sent to the server when the player shoots or moves the rocket
- A message is received from the server when it joins the game and when another player shoots, moves it rocket, arrives, or departs
- The server receives messages from a player, manages the joining of new players, and manages the departure of players
- When message arrives from a player, the server broadcasts the message to the other players
- When a player joins the game the server provides it with its starting world and broadcasts a message to all players that they have a new `ally`
- The starting world depends on when $\texttt{Player}_i$ arrives
- $\texttt{Player}_i$ is the first player: sends the initial `world`
- Otherwise: server requests `world` existing player and sends it to $\texttt{Player}_i$
- In order to properly select recipients of messages each player component must have a distinct name
- The server rejects any new player that has a name already associated with an existing player

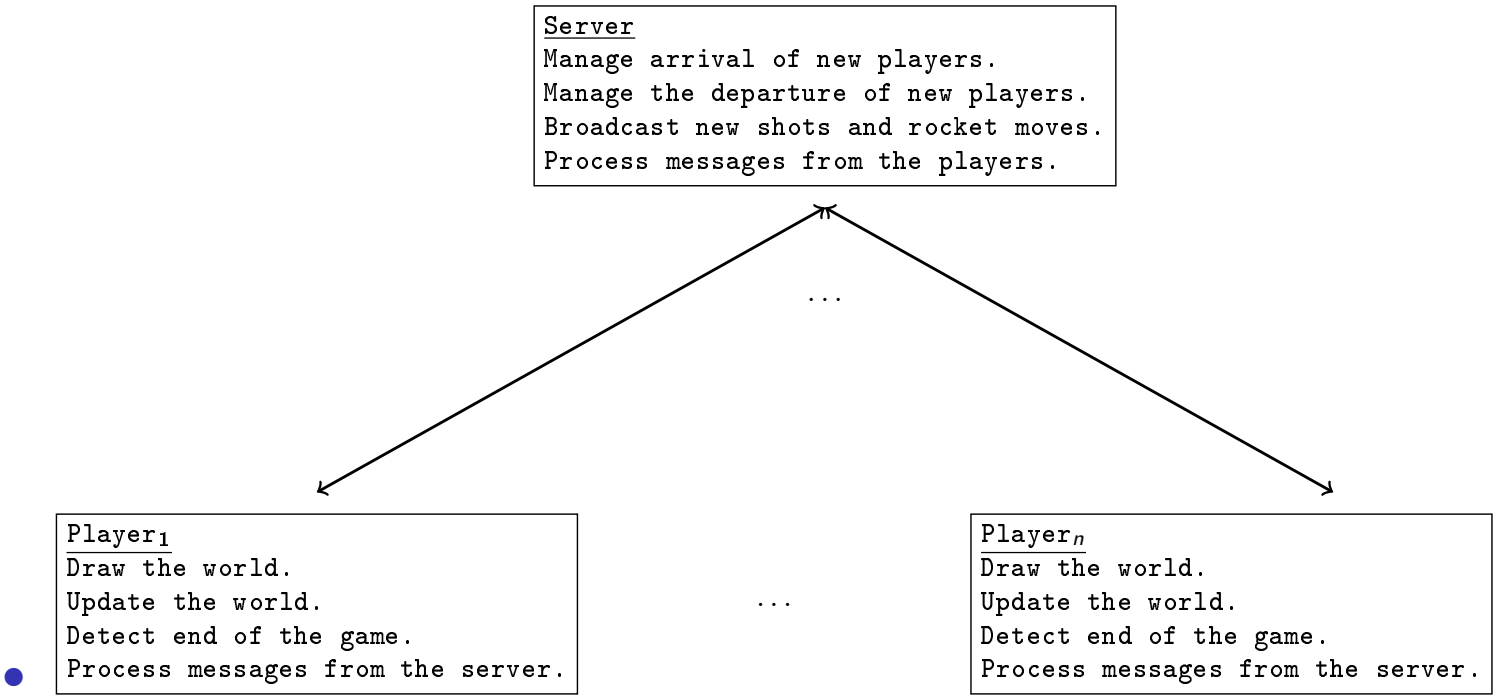
Aliens Attack Version 7

Components

- Aliens Attack 6 player's program: draw, detect game over, and process key events, clock ticks, and messages from the server
- Intuitive design idea: each player component performs the same tasks
- Each player component sends a message to a server when a player induced change occurs
- A message is sent to the server when the player shoots or moves the rocket
- A message is received from the server when it joins the game and when another player shoots, moves it rocket, arrives, or departs
- The server receives messages from a player, manages the joining of new players, and manages the departure of players
- When message arrives from a player, the server broadcasts the message to the other players
- When a player joins the game the server provides it with its starting world and broadcasts a message to all players that they have a new `ally`
- The starting world depends on when $\texttt{Player}_i$ arrives
- $\texttt{Player}_i$ is the first player: sends the initial `world`
- Otherwise: server requests `world` existing player and sends it to $\texttt{Player}_i$
- In order to properly select recipients of messages each player component must have a distinct name
- The server rejects any new player that has a name already associated with an existing player
- When a player departs the game the server sends a message to all the other players to remove the corresponding `ally`

Aliens Attack Version 7

Components



Aliens Attack Version 7

Data Definitions

- In order to send messages to specific players the server needs to track the players in the game
- Given that each player is represented as an `iworld` in our `API` the universe is defined as follows:

```
;; A universe is a (listof iworld), where each iworld
;; has a unique name

;; Sample instances of universe
(define INIT-UNIV '())
(define A-UNIV    (list iworld1 iworld2))
```

Aliens Attack Version 7

Communication Protocol

- Communication chains are sparked by game-changing events that must be communicated
- Think carefully about what events cause a change
- What are these for players?

Aliens Attack Version 7

Communication Protocol

- Communication chains are sparked by game-changing events that must be communicated
- Think carefully about what events cause a change
- What are these for players?
- Player shoots
- Player moves her rocket

Aliens Attack Version 7

Communication Protocol

- Communication chains are sparked by game-changing events that must be communicated
- Think carefully about what events cause a change
- What are these for players?
- Player shoots
- Player moves her rocket
- What events cause the server to start a communication chain?

Aliens Attack Version 7

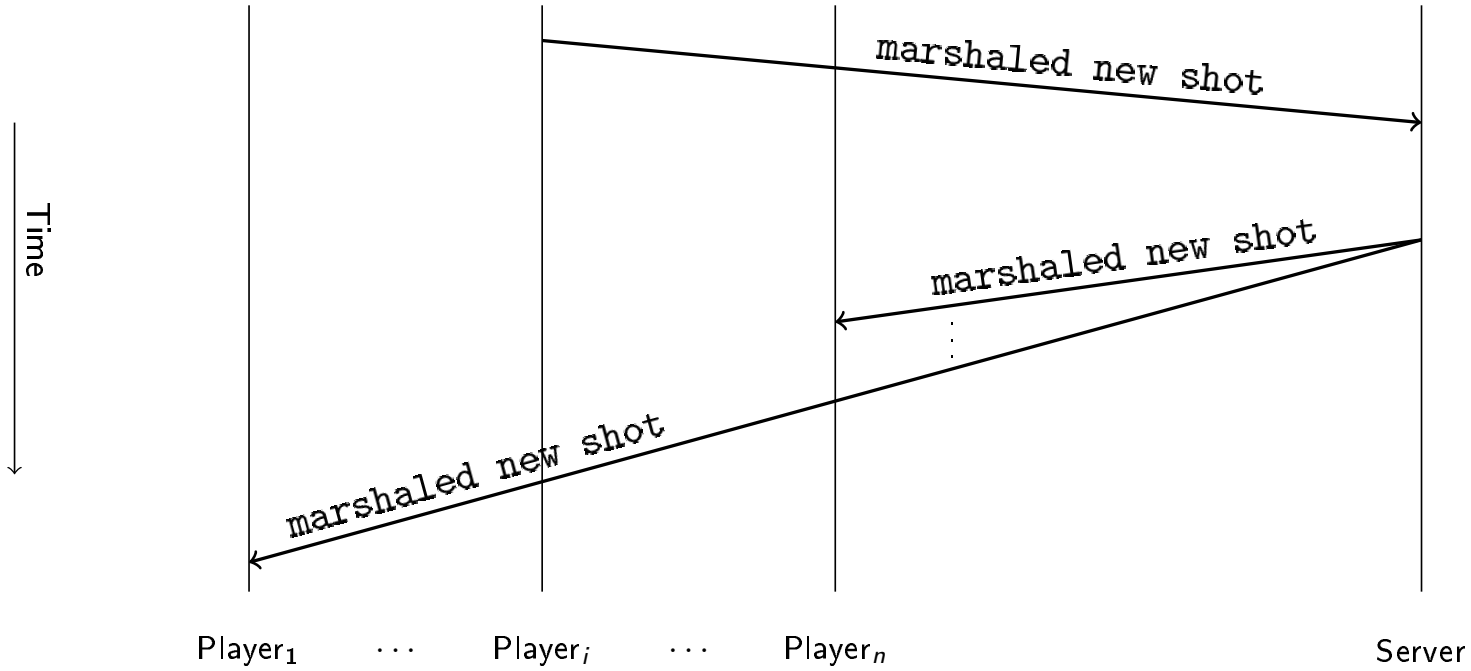
Communication Protocol

- Communication chains are sparked by game-changing events that must be communicated
- Think carefully about what events cause a change
- What are these for players?
- Player shoots
- Player moves her rocket
- What events cause the server to start a communication chain?
- Player joins
- Player leaves

Aliens Attack Version 7

Communication Protocol

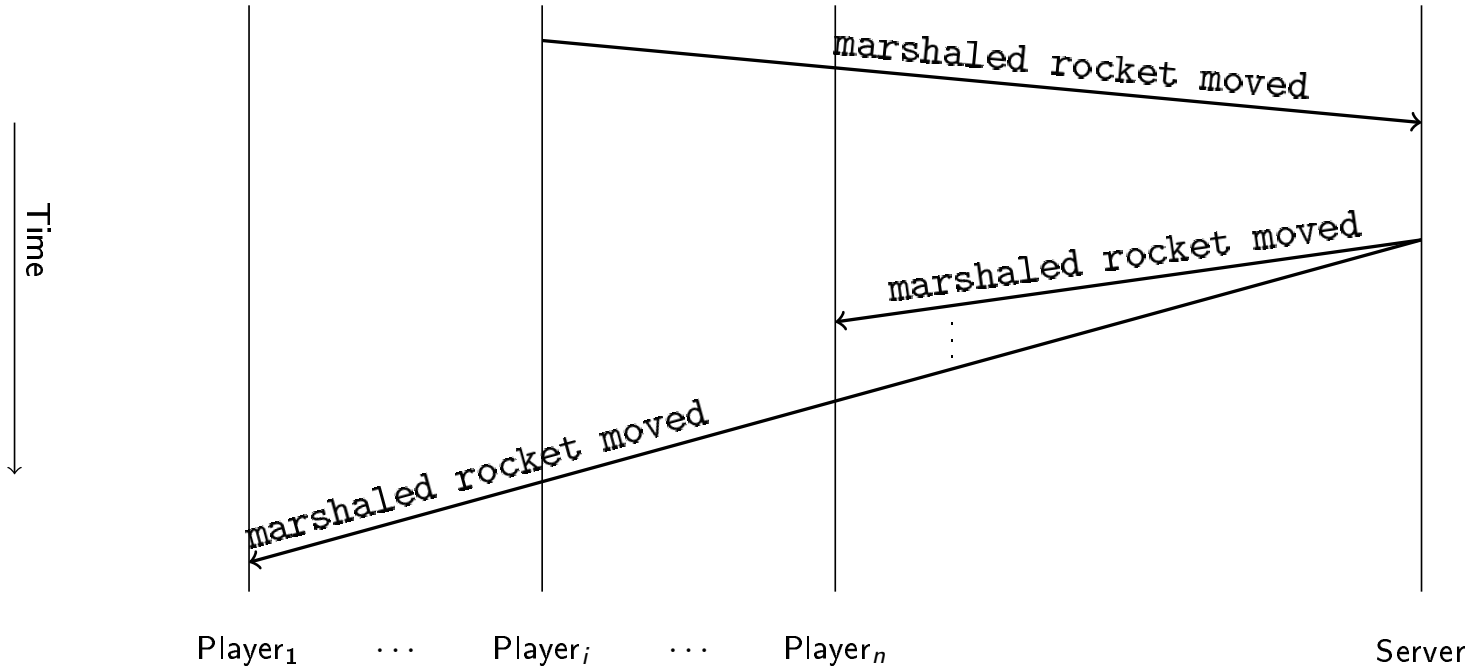
- Player i shoots



Aliens Attack Version 7

Communication Protocol

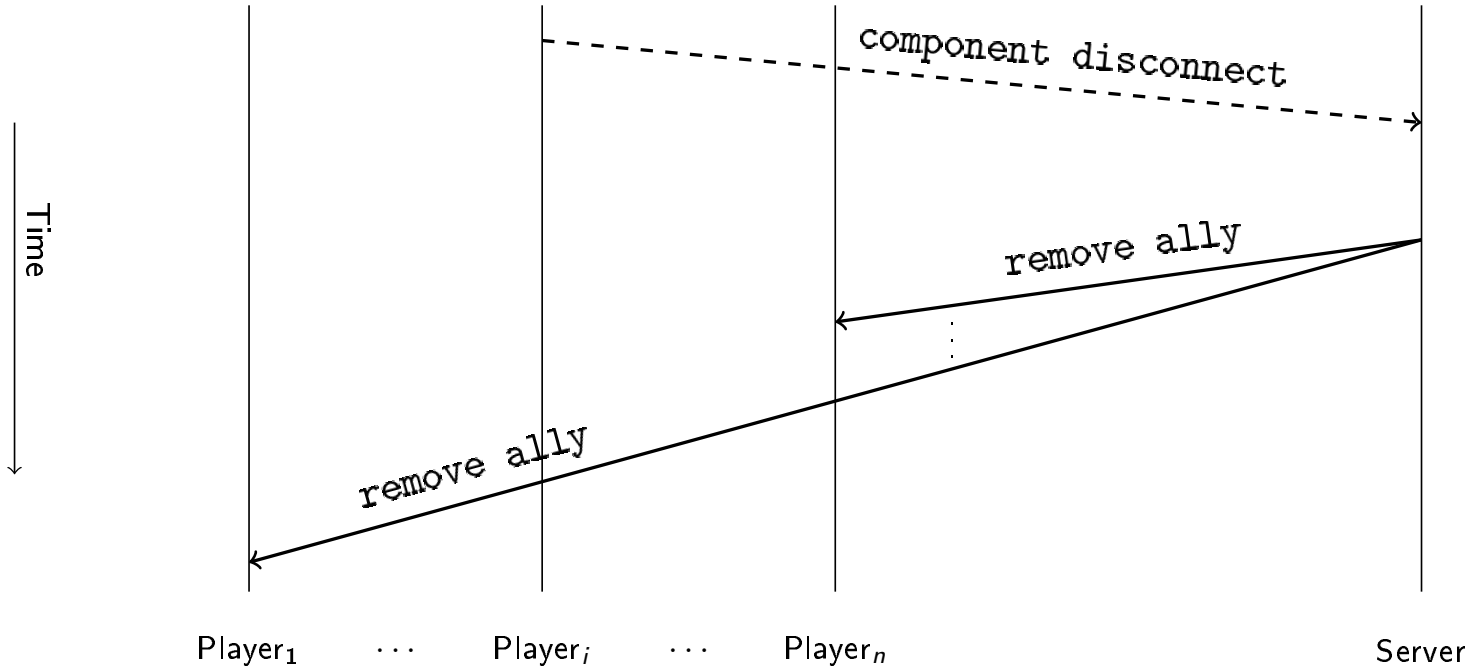
- Rocket move by player `i`



Aliens Attack Version 7

Communication Protocol

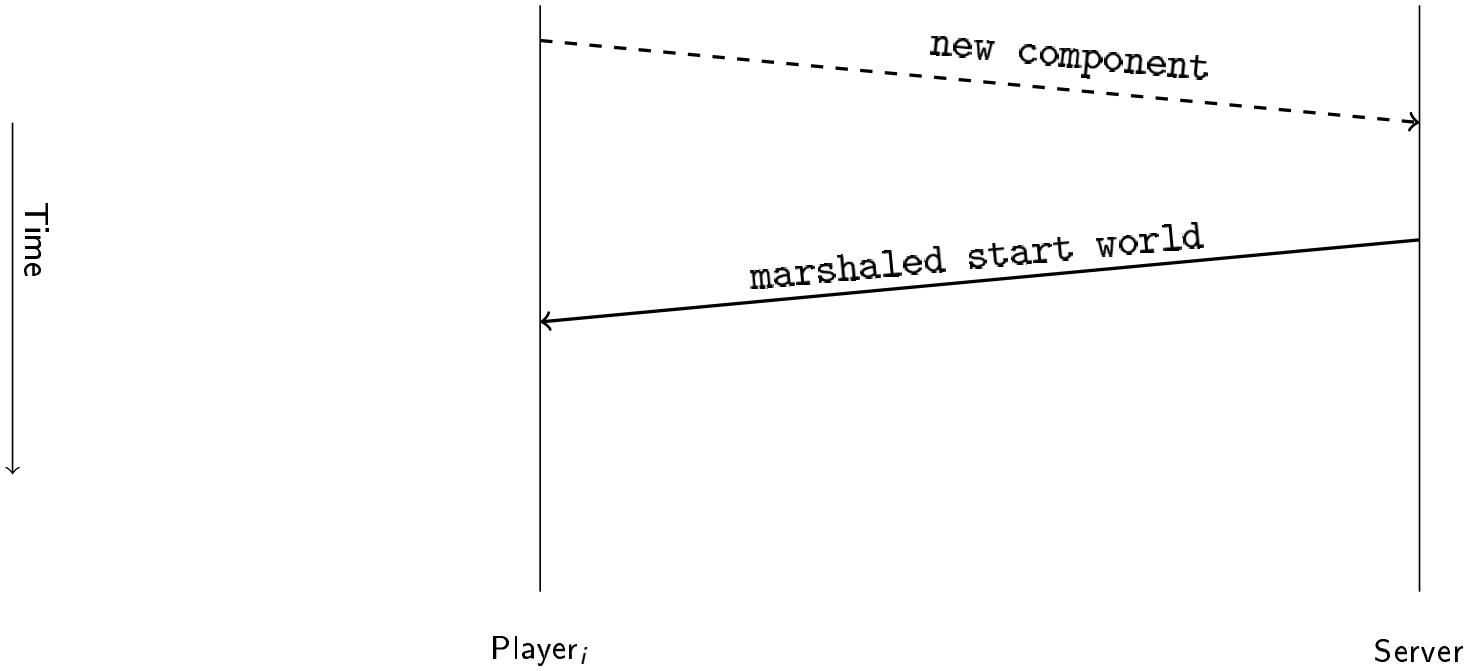
- Player i leaves the game



Aliens Attack Version 7

Communication Protocol

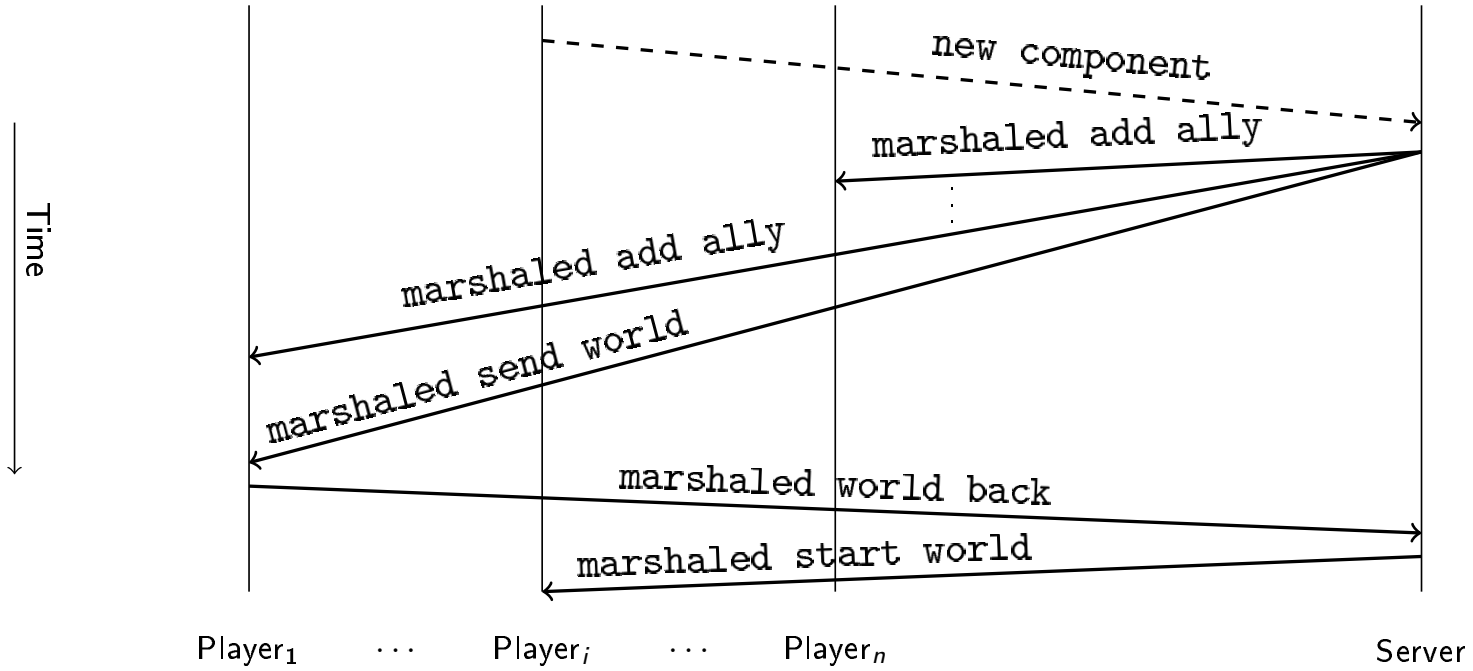
- Player i joins an empty universe



Aliens Attack Version 7

Communication Protocol

- Player i joins a nonempty universe



Aliens Attack Version 7

Marshaled-Data Definitions

```
;;  A marshaled ally (mr) is a (list image-x string)
;;  A marshaled alien (ma) is a (list image-x image-y)
;;  A marshaled shot (ms) is either
;;    1. (list image-x image-y)
;;    2. 'no-shot
;;  A marshaled world (mw) is a (list (listof mr) (listof ma) direction (listof ms))
```

Aliens Attack Version 7

Marshaled-Data Definitions

```
;;  A marshaled ally (mr) is a (list image-x string)
;;  A marshaled alien (ma) is a (list image-x image-y)
;;  A marshaled shot (ms) is either
;;    1. (list image-x image-y)
;;    2. 'no-shot
;;  A marshaled world (mw) is a (list (listof mr) (listof ma) direction (listof ms))
```

```
#| Templates in textbook |#
(define MR1 '(10 "iworld1")) (define MR2 '(8  "iworld3")) (define MR3 `(11 ,MY-NAME))
(define MALLY1 (list 10 "Rolando"))  (define MALLY2  (list 10 "Margarita"))
(define MA '(14 3)) (define MALIEN1 (list 0 7)) (define MALIEN2 (list 9 4))
(define MS1 NO-SHOT)     (define MS2 '(2  2))
(define MSHOT1  NO-SHOT) (define MSHOT2 (list 11 9))
(define MLOS   '((2 2)))
(define MW '(((7 "iworld1") (3 "iworld2"))
             ((15 2))
             'left
             ((8 5) (7 2))))
(define MWORLD1
  (list (list (list 16 "Cristian") (list 7  "Laura") (list 6  "Walter"))
        (list (list 13 4) (list 11 11))
        'right
        (list (list 2  3) (list 12 8) (list 5  14))))
(define MWORLD2
  (list (list (list 0 "Marce") (list 8 "Chaty") (list 4 "Maggie") (list 2 "Christy"))
        '()
        'right
        '()))
```

Aliens Attack Version 7

Message Data Definitions

- ```
  ;; A to-player message (tpm) is either
  ;;  1. (list 'rckt-move  ma)
  ;;  2. (list 'new-shot   ms)
  ;;  3. (list 'new-ally   ma)
  ;;  4. (list 'rm-ally    string)
  ;;  5. (list 'send-world string)
  ;;  6. (cons 'start      mw)
  ```
- Template in textbook
- ```
  (define RM-MSG  (list 'rckt-move   MR2))
  (define RM-MSG2 (list 'rckt-move   MR3))
  (define NS-MSG  (list 'new-shot    MS1))
  (define NS-MSG2 (list 'new-shot    MS2))
  (define NA-MSG  (list 'new-ally    MR2))
  (define RA-MSG  (list 'rm-ally     "world1"))
  (define RA-MSG2 (list 'rm-ally     "Margarita"))
  (define SW-MSG  (list 'send-world  "world2"))
  (define SW-MSG2  (list 'send-world "Fernando"))
  (define ST-MSG  (cons 'start       MW))
  ```

Aliens Attack Version 7

Message Data Definitions

- ```
  ;; A to-player message (tpm) is either
  ;;  1. (list 'rckt-move  ma)
  ;;  2. (list 'new-shot   ms)
  ;;  3. (list 'new-ally   ma)
  ;;  4. (list 'rm-ally    string)
  ;;  5. (list 'send-world string)
  ;;  6. (cons 'start      mw)
  ```
- Template in textbook
- ```
  (define RM-MSG  (list 'rckt-move   MR2))
  (define RM-MSG2 (list 'rckt-move   MR3))
  (define NS-MSG  (list 'new-shot    MS1))
  (define NS-MSG2 (list 'new-shot    MS2))
  (define NA-MSG  (list 'new-ally    MR2))
  (define RA-MSG  (list 'rm-ally     "world1"))
  (define RA-MSG2 (list 'rm-ally     "Margarita"))
  (define SW-MSG  (list 'send-world  "world2"))
  (define SW-MSG2  (list 'send-world "Fernando"))
  (define ST-MSG  (cons 'start       MW))
  ```
- ```
  ;; A to-server message (tsm) is either
  ;;  1. (list 'rckt-move ma)
  ;;  2. (list 'new-shot  ms)
  ;;  3. (cons 'world-back (cons string mw))
  ```
- Template in textbook
- ```
  ;; Sample instances of tsm
  (define SRM-MSG RM-MSG)
  (define SNS-MSG NS-MSG)
  (define SWB-MSG (cons 'world-back (cons "iworld2" MW)))
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

- Marshalling and unmarshalling, respectively, make data unfit for transmission into data that is fit for transmission and back
- Four varieties of marshaled data: marshaled ally (`mr`), marshaled alien (`ma`), marshaled shot (`ms`), and marshaled world (`mw`)
- A list of `mr`, a list of `ma`, and a list of `ms` are also defined as part of an `mw`
- We shall develop marshalling and unmarshalling functions for each of these types

Aliens Attack Version 7

Marshalling and Unmarshalling

- To marshal an ally you need an `ally` as input and you need to output an `mr`

Aliens Attack Version 7

Marshalling and Unmarshalling

- To marshal an ally you need an ally as input and you need to output an mr
-
```
;; Sample expressions for marshal-ally
(define M-ALLY1 (list (ally-rocket INIT-ALLY) (ally-name INIT-ALLY)))
(define M-ALLY2 (list (ally-rocket INIT-ALLY2) (ally-name INIT-ALLY2)))

;; ally → mr
;; Purpose: Marshal the given ally
(define (marshal-ally an-ally)
  (list (ally-rocket an-ally) (ally-name an-ally)))

;; Tests using sample computations for marshal-ally
(check-expect (marshal-ally  INIT-ALLY)  M-ALLY1)
(check-expect (marshal-ally  INIT-ALLY2) M-ALLY2)
;; Tests using sample values for marshal-ally
(check-expect (marshal-ally  (make-ally 12 "Cordula")) (list 12 "Cordula"))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

- To marshal an ally you need an ally as input and you need to output an mr
- ```
  ;; Sample expressions for marshal-ally
  (define M-ALLY1 (list (ally-rocket INIT-ALLY) (ally-name INIT-ALLY)))
  (define M-ALLY2 (list (ally-rocket INIT-ALLY2) (ally-name INIT-ALLY2)))

  ;; ally → mr
  ;; Purpose: Marshal the given ally
  (define (marshal-ally an-ally)
    (list (ally-rocket an-ally) (ally-name an-ally)))

  ;; Tests using sample computations for marshal-ally
  (check-expect (marshal-ally  INIT-ALLY)  M-ALLY1)
  (check-expect (marshal-ally  INIT-ALLY2) M-ALLY2)
  ;; Tests using sample values for marshal-ally
  (check-expect (marshal-ally  (make-ally 12 "Cordula")) (list 12 "Cordula"))
  ```
- The unmarshalling function must take as input an mr and return an ally:

  ```
  ;; Sample expressions for unmarshal-ally
  (define UALLY1 (make-ally (first  MALLY1) (second MALLY1)))
  (define UALLY2 (make-ally (first  MALLY2) (second MALLY2)))

  ;; mr → ally
  ;; Purpose: Unmarshal the given marshaled ally
  (define (unmarshal-ally ma)
    (local [(define rocket (first ma))
            (define name   (second ma))]
      (make-ally rocket name)))

  ;; Tests using sample computations for unmarshal-ally
  (check-expect (unmarshal-ally MALLY1) UALLY1)
  (check-expect (unmarshal-ally MALLY2) UALLY2)
  ;; Tests using sample values for unmarshal-ally
  (check-expect (unmarshal-ally (list 12 "Cordula")) (make-ally 12 "Cordula"))
  ```
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

- ```
  ;; Sample expressions for marshal-alien
  (define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
  ```
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; alien → ma
;; Purpose: Marshal the given alien
(define (marshal-alien an-alien)
```

```
;; Sample expressions for marshal-alien
(define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

- ```
  ;; alien → ma
  ;; Purpose: Marshal the given alien
  (define (marshal-alien an-alien)
  ```

- ```
  ;; Sample expressions for marshal-alien
  (define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
  ```

- ```
  ;; Tests using sample computations for marshal-alien
  (check-expect (marshal-alien INIT-ALIEN2) MINIT-ALIEN2)

  ;; Tests using sample values for unmarshal-alien
  (check-expect (marshal-alien (make-posn 7 2)) (list 7 2))
  ```
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; alien → ma
;; Purpose: Marshal the given alien
(define (marshal-alien an-alien)
  (list (posn-x an-alien) (posn-y an-alien)))
```

```
;; Sample expressions for marshal-alien
(define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
```

```
;; Tests using sample computations for marshal-alien
(check-expect (marshal-alien INIT-ALIEN2) MINIT-ALIEN2)

;; Tests using sample values for unmarshal-alien
(check-expect (marshal-alien (make-posn 7 2)) (list 7 2))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; alien → ma
;; Purpose: Marshal the given alien
(define (marshal-alien an-alien)
  (list (posn-x an-alien) (posn-y an-alien)))
```

```
;; Sample expressions for marshal-alien
(define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
```

```
;; Tests using sample computations for marshal-alien
(check-expect (marshal-alien INIT-ALIEN2) MINIT-ALIEN2)

;; Tests using sample values for unmarshal-alien
(check-expect (marshal-alien (make-posn 7 2)) (list 7 2))
```

```
;; Sample expressions for unmarshal-alien
(define UALIEN2 (make-posn (first MINIT-ALIEN2) (second MINIT-ALIEN2)))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; alien → ma
;; Purpose: Marshal the given alien
(define (marshal-alien an-alien)
  (list (posn-x an-alien) (posn-y an-alien)))
```

```
;; Sample expressions for marshal-alien
(define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
```

```
;; Tests using sample computations for marshal-alien
(check-expect (marshal-alien INIT-ALIEN2) MINIT-ALIEN2)

;; Tests using sample values for unmarshal-alien
(check-expect (marshal-alien (make-posn 7 2)) (list 7 2))
```

```
;; ma → alien
;; Purpose: Unmarshal the given marshaled alien
(define (unmarshal-alien ma)
```

```
;; Sample expressions for unmarshal-alien
(define UALIEN2 (make-posn (first MINIT-ALIEN2) (second MINIT-ALIEN2)))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; alien → ma
;; Purpose: Marshal the given alien
(define (marshal-alien an-alien)
  (list (posn-x an-alien) (posn-y an-alien)))
```

```
;; Sample expressions for marshal-alien
(define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
```

```
;; Tests using sample computations for marshal-alien
(check-expect (marshal-alien INIT-ALIEN2) MINIT-ALIEN2)

;; Tests using sample values for unmarshal-alien
(check-expect (marshal-alien (make-posn 7 2)) (list 7 2))
```

```
;; ma → alien
;; Purpose: Unmarshal the given marshaled alien
(define (unmarshal-alien ma)
```

```
;; Sample expressions for unmarshal-alien
(define UALIEN2 (make-posn (first MINIT-ALIEN2) (second MINIT-ALIEN2)))
```

```
;; Tests using sample computations for unmarshal-alien
(check-expect (unmarshal-alien MINIT-ALIEN2) UALIEN2)

;; Tests using sample values for unmarshal-alien
(check-expect (unmarshal-alien (list 7 2)) (make-posn 7 2))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; alien → ma
;; Purpose: Marshal the given alien
(define (marshal-alien an-alien)
  (list (posn-x an-alien) (posn-y an-alien)))
```

```
;; Sample expressions for marshal-alien
(define MINIT-ALIEN2 (list (posn-x INIT-ALIEN2) (posn-y INIT-ALIEN2)))
```

```
;; Tests using sample computations for marshal-alien
(check-expect (marshal-alien INIT-ALIEN2) MINIT-ALIEN2)

;; Tests using sample values for unmarshal-alien
(check-expect (marshal-alien (make-posn 7 2)) (list 7 2))
```

```
;; ma → alien
;; Purpose: Unmarshal the given marshaled alien
(define (unmarshal-alien ma)
  (local [(define img-x (first ma))
          (define img-y (second ma))]
    (make-posn img-x img-y)))
```

```
;; Sample expressions for unmarshal-alien
(define UALIEN2 (make-posn (first MINIT-ALIEN2) (second MINIT-ALIEN2)))
```

```
;; Tests using sample computations for unmarshal-alien
(check-expect (unmarshal-alien MINIT-ALIEN2) UALIEN2)

;; Tests using sample values for unmarshal-alien
(check-expect (unmarshal-alien (list 7 2)) (make-posn 7 2))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

- ```
  ;; Sample expressions for marshal-shot
  (define M-NO-SHOT NO-SHOT)
  (define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
  ```
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

- ```
  ;; shot → ms
  ;; Purpose: Marshal the given shot
  (define (marshal-shot a-shot)
  ```

- ```
  ;; Sample expressions for marshal-shot
  (define M-NO-SHOT NO-SHOT)
  (define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

- ```
  ;; shot → ms
  ;; Purpose: Marshal the given shot
  (define (marshal-shot a-shot)
  ```

- ```
  ;; Sample expressions for marshal-shot
  (define M-NO-SHOT NO-SHOT)
  (define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
  ```

- ```
  ;; Tests using sample computations for marshal-shot
  (check-expect (marshal-shot NO-SHOT) M-NO-SHOT)
  (check-expect (marshal-shot SHOT2)   M-SHOT2)
  ;; Tests using sample values for marshal-shot
  (check-expect (marshal-shot (make-posn 2 2)) (list 2 2))
  ```
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; shot → ms
;; Purpose: Marshal the given shot
(define (marshal-shot a-shot)
  (if (eq? a-shot NO-SHOT)
      NO-SHOT
      (list (posn-x a-shot) (posn-y a-shot))))
```

```
;; Sample expressions for marshal-shot
(define M-NO-SHOT NO-SHOT)
(define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
```

```
;; Tests using sample computations for marshal-shot
(check-expect (marshal-shot NO-SHOT) M-NO-SHOT)
(check-expect (marshal-shot SHOT2)   M-SHOT2)
;; Tests using sample values for marshal-shot
(check-expect (marshal-shot (make-posn 2 2)) (list 2 2))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; shot → ms
;; Purpose: Marshal the given shot
(define (marshal-shot a-shot)
  (if (eq? a-shot NO-SHOT)
      NO-SHOT
      (list (posn-x a-shot) (posn-y a-shot))))
```

```
;; Sample expressions for marshal-shot
(define M-NO-SHOT NO-SHOT)
(define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
```

```
;; Tests using sample computations for marshal-shot
(check-expect (marshal-shot NO-SHOT) M-NO-SHOT)
(check-expect (marshal-shot SHOT2)   M-SHOT2)
;; Tests using sample values for marshal-shot
(check-expect (marshal-shot (make-posn 2 2)) (list 2 2))
```

```
;; Sample expressions for unmarshal-shot
(define USHOT1 NO-SHOT)
(define USHOT2 (make-posn (first MSHOT2) (second MSHOT2)))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; shot → ms
;; Purpose: Marshal the given shot
(define (marshal-shot a-shot)
  (if (eq? a-shot NO-SHOT)
      NO-SHOT
      (list (posn-x a-shot) (posn-y a-shot))))
```

```
;; Sample expressions for marshal-shot
(define M-NO-SHOT NO-SHOT)
(define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
```

```
;; Tests using sample computations for marshal-shot
(check-expect (marshal-shot NO-SHOT) M-NO-SHOT)
(check-expect (marshal-shot SHOT2)   M-SHOT2)
;; Tests using sample values for marshal-shot
(check-expect (marshal-shot (make-posn 2 2)) (list 2 2))
```

```
;; ms → shot
;; Purpose: Unmarshal the given marshaled shot
(define (unmarshal-shot ms)
```

```
;; Sample expressions for unmarshal-shot
(define USHOT1 NO-SHOT)
(define USHOT2 (make-posn (first MSHOT2) (second MSHOT2)))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; shot → ms
;; Purpose: Marshal the given shot
(define (marshal-shot a-shot)
  (if (eq? a-shot NO-SHOT)
      NO-SHOT
      (list (posn-x a-shot) (posn-y a-shot))))
```

```
;; Sample expressions for marshal-shot
(define M-NO-SHOT NO-SHOT)
(define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
```

```
;; Tests using sample computations for marshal-shot
(check-expect (marshal-shot NO-SHOT) M-NO-SHOT)
(check-expect (marshal-shot SHOT2)   M-SHOT2)
;; Tests using sample values for marshal-shot
(check-expect (marshal-shot (make-posn 2 2)) (list 2 2))
```

```
;; ms → shot
;; Purpose: Unmarshal the given marshaled shot
(define (unmarshal-shot ms)
```

```
;; Sample expressions for unmarshal-shot
(define USHOT1 NO-SHOT)
(define USHOT2 (make-posn (first MSHOT2) (second MSHOT2)))
```

```
;; Tests using sample computations for unmarshal-shot
(check-expect (unmarshal-shot NO-SHOT) USHOT1)
(check-expect (unmarshal-shot MSHOT2) USHOT2)
;; Tests using sample values for unmarshal-shot
(check-expect (unmarshal-shot (list 2 2)) (make-posn 2 2))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; shot → ms
;; Purpose: Marshal the given shot
(define (marshal-shot a-shot)
  (if (eq? a-shot NO-SHOT)
      NO-SHOT
      (list (posn-x a-shot) (posn-y a-shot))))
```

```
;; Sample expressions for marshal-shot
(define M-NO-SHOT NO-SHOT)
(define M-SHOT2   (list (posn-x SHOT2) (posn-y SHOT2)))
```

```
;; Tests using sample computations for marshal-shot
(check-expect (marshal-shot NO-SHOT) M-NO-SHOT)
(check-expect (marshal-shot SHOT2)   M-SHOT2)
;; Tests using sample values for marshal-shot
(check-expect (marshal-shot (make-posn 2 2)) (list 2 2))
```

```
;; ms → shot
;; Purpose: Unmarshal the given marshaled shot
(define (unmarshal-shot ms)
  (if (list? ms)
      (local [(define img-x (first ms))
              (define img-y (second ms))]
        (make-posn img-x img-y))
      ms))
```

```
;; Sample expressions for unmarshal-shot
(define USHOT1 NO-SHOT)
(define USHOT2 (make-posn (first MSHOT2) (second MSHOT2)))
```

```
;; Tests using sample computations for unmarshal-shot
(check-expect (unmarshal-shot NO-SHOT) USHOT1)
(check-expect (unmarshal-shot MSHOT2) USHOT2)
;; Tests using sample values for unmarshal-shot
(check-expect (unmarshal-shot (list 2 2)) (make-posn 2 2))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; Sample expressions for marshal-world
(define MWORLD3 (list (map marshal-ally (world-allies INIT-WORLD))
                      (map marshal-alien (world-aliens INIT-WORLD))
                      (world-dir INIT-WORLD)
                      (map marshal-shot (world-shots INIT-WORLD))))
(define MWORLD4 (list (map marshal-ally (world-allies INIT-WORLD2))
                      (map marshal-alien (world-aliens INIT-WORLD2))
                      (world-dir INIT-WORLD2)
                      (map marshal-shot (world-shots INIT-WORLD2))))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; world → mw  Purpose: Marshal the given world
;; ASSUMPTION: The given world is a structure
(define (marshal-world a-world)
```

```
;; Sample expressions for marshal-world
(define MWORLD3 (list (map marshal-ally (world-allies INIT-WORLD))
                      (map marshal-alien (world-aliens INIT-WORLD))
                      (world-dir INIT-WORLD)
                      (map marshal-shot (world-shots INIT-WORLD))))
(define MWORLD4 (list (map marshal-ally (world-allies INIT-WORLD2))
                      (map marshal-alien (world-aliens INIT-WORLD2))
                      (world-dir INIT-WORLD2)
                      (map marshal-shot (world-shots INIT-WORLD2))))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

- ```
  ;; world → mw  Purpose: Marshal the given world
  ;; ASSUMPTION: The given world is a structure
  (define (marshal-world a-world)
  ```

- ```
  ;; Sample expressions for marshal-world
  (define MWORLD3 (list (map marshal-ally (world-allies INIT-WORLD))
                        (map marshal-alien (world-aliens INIT-WORLD))
                        (world-dir INIT-WORLD)
                        (map marshal-shot (world-shots INIT-WORLD))))
  (define MWORLD4 (list (map marshal-ally (world-allies INIT-WORLD2))
                        (map marshal-alien (world-aliens INIT-WORLD2))
                        (world-dir INIT-WORLD2)
                        (map marshal-shot (world-shots INIT-WORLD2))))
  ```

- ```
  ;; Tests using sample computations for marshal-world
  (check-expect (marshal-world INIT-WORLD)  MWORLD3)
  (check-expect (marshal-world INIT-WORLD2) MWORLD4)
  ;; Tests using sample values for marshal-world
  (check-expect
    (marshal-world (make-world (list (make-ally 5 "Luis") (make-ally 8 "Cova"))
                               (list (make-posn  2 17))
                               "left"
                               '()))
    (list (list (list 5 "Luis") (list 8 "Cova"))
          (list (list  2 17))
          "left"
          '()))
  ```
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; world → mw  Purpose: Marshal the given world
;; ASSUMPTION: The given world is a structure
(define (marshal-world a-world)
  (list (map marshal-ally  (world-allies a-world))
        (map marshal-alien (world-aliens a-world))
        (world-dir a-world)
        (map marshal-shot (world-shots a-world))))
```

```
;; Sample expressions for marshal-world
(define MWORLD3 (list (map marshal-ally (world-allies INIT-WORLD))
                      (map marshal-alien (world-aliens INIT-WORLD))
                      (world-dir INIT-WORLD)
                      (map marshal-shot (world-shots INIT-WORLD))))
(define MWORLD4 (list (map marshal-ally (world-allies INIT-WORLD2))
                      (map marshal-alien (world-aliens INIT-WORLD2))
                      (world-dir INIT-WORLD2)
                      (map marshal-shot (world-shots INIT-WORLD2))))
```

```
;; Tests using sample computations for marshal-world
(check-expect (marshal-world INIT-WORLD)  MWORLD3)
(check-expect (marshal-world INIT-WORLD2) MWORLD4)
;; Tests using sample values for marshal-world
(check-expect
  (marshal-world (make-world (list (make-ally 5 "Luis") (make-ally 8 "Cova"))
                             (list (make-posn  2 17))
                             "left"
                             '()))
  (list (list (list 5 "Luis") (list 8 "Cova"))
        (list (list  2 17))
        "left"
        '()))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

- ```
  ;; Sample expressions for unmarshal-world Silly repetition in textbook
  (define UWORLD1 (make-world (map unmarshal-ally  (first MWORLD1))
                              (map unmarshal-alien (second MWORLD1))
                              (third MWORLD1)
                              (map unmarshal-shot (fourth MWORLD1))))
  (define UWORLD2 (make-world (map unmarshal-ally  (first MWORLD2))
                              (map unmarshal-alien (second MWORLD2))
                              (third MWORLD2)
                              (map unmarshal-shot (fourth MWORLD2))))
  ```
  ```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; mw → world  Purpose: Unmarshal the given world
(define (unmarshal-world mw)
```

```
;; Sample expressions for unmarshal-world Silly repetition in textbook
(define UWORLD1 (make-world (map unmarshal-ally  (first MWORLD1))
                            (map unmarshal-alien (second MWORLD1))
                            (third MWORLD1)
                            (map unmarshal-shot (fourth MWORLD1))))
(define UWORLD2 (make-world (map unmarshal-ally  (first MWORLD2))
                            (map unmarshal-alien (second MWORLD2))
                            (third MWORLD2)
                            (map unmarshal-shot (fourth MWORLD2))))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; mw → world  Purpose: Unmarshal the given world
(define (unmarshal-world mw)
```

```
;; Sample expressions for unmarshal-world Silly repetition in textbook
(define UWORLD1 (make-world (map unmarshal-ally  (first MWORLD1))
                            (map unmarshal-alien (second MWORLD1))
                            (third MWORLD1)
                            (map unmarshal-shot (fourth MWORLD1))))
(define UWORLD2 (make-world (map unmarshal-ally  (first MWORLD2))
                            (map unmarshal-alien (second MWORLD2))
                            (third MWORLD2)
                            (map unmarshal-shot (fourth MWORLD2))))
```

```
;; Tests using sample computations for unmarshal-world
(check-expect (unmarshal-world MWORLD1) UWORLD1)
(check-expect (unmarshal-world MWORLD2) UWORLD2)
;; Tests using sample values for unmarshal-world
(check-expect
 (unmarshal-world (list (list (list 5 "Neil") (list 6 "Constance")
                              (list 7 "Madrid") (list 8 "Skyler"))
                        (list (list 0 4))
                        "down"
                        '()))
 (make-world (list (make-ally 5 "Neil") (make-ally 6 "Constance")
                   (make-ally 7 "Madrid") (make-ally 8 "Skyler"))
             (list (make-posn 0 4))
             "down"
             '()))
```

Aliens Attack Version 7

Marshalling and Unmarshalling

```
;; mw → world  Purpose: Unmarshal the given world
(define (unmarshal-world mw)
  (local [(define lomr (first mw)) (define loma (second mw))
          (define dir  (third mw)) (define loms (fourth mw))]
    (make-world (map unmarshal-ally  lomr)
                (map unmarshal-alien loma)
                dir
                (map unmarshal-shot loms))))
```

```
;; Sample expressions for unmarshal-world Silly repetition in textbook
(define UWORLD1 (make-world (map unmarshal-ally  (first MWORLD1))
                            (map unmarshal-alien (second MWORLD1))
                            (third MWORLD1)
                            (map unmarshal-shot (fourth MWORLD1))))
(define UWORLD2 (make-world (map unmarshal-ally  (first MWORLD2))
                            (map unmarshal-alien (second MWORLD2))
                            (third MWORLD2)
                            (map unmarshal-shot (fourth MWORLD2))))
```

```
;; Tests using sample computations for unmarshal-world
(check-expect (unmarshal-world MWORLD1) UWORLD1)
(check-expect (unmarshal-world MWORLD2) UWORLD2)
;; Tests using sample values for unmarshal-world
(check-expect
 (unmarshal-world (list (list (list 5 "Neil") (list 6 "Constance")
                              (list 7 "Madrid") (list 8 "Skyler"))
                        (list (list 0 4))
                        "down"
                        '()))
 (make-world (list (make-ally 5 "Neil") (make-ally 6 "Constance")
                   (make-ally 7 "Madrid") (make-ally 8 "Skyler"))
             (list (make-posn 0 4))
             "down"
             '()))
```

Aliens Attack Version 7

Component Implementation: Player

- All the new data definitions are associated with the communication protocol
- All code refinement focuses on functions that send messages to the server and on the function to process `tpms`

Aliens Attack Version 7

Component Implementation: Player

- ```
  ;; world key → world or package
  ;; Purpose: Return new package or world after a key event
  (define (process-key a-world a-key)
    (cond [(key=? a-key "right")
           (local
             [(define nw (make-world (move-ally-right
                                      MY-NAME (world-allies a-world))
                                     (world-aliens a-world)
                                     (world-dir a-world)
                                     (world-shots a-world)))
              (define na (get-ally MY-NAME (world-allies nw)))]
             (make-package nw (list 'rckt-move (marshal-ally na))))]
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

```
;; world key → world or package
;; Purpose: Return new package or world after a key event
(define (process-key a-world a-key)
  (cond [(key=? a-key "right")
         (local
           [(define nw (make-world (move-ally-right
                                    MY-NAME (world-allies a-world))
                                   (world-aliens a-world)
                                   (world-dir a-world)
                                   (world-shots a-world)))
            (define na (get-ally MY-NAME (world-allies nw)))]
           (make-package nw (list 'rckt-move (marshal-ally na))))]
        [(key=? a-key "left")
         (local
           [(define nw (make-world (move-ally-left MY-NAME
                                                   (world-allies a-world))
                                   (world-aliens a-world)
                                   (world-dir a-world)
                                   (world-shots a-world)))
            (define na (get-ally MY-NAME (world-allies nw)))]
           (make-package nw (list 'rckt-move (marshal-ally na))))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world key → world or package
;; Purpose: Return new package or world after a key event
(define (process-key a-world a-key)
  (cond [(key=? a-key "right")
         (local
           [(define nw (make-world (move-ally-right
                                    MY-NAME (world-allies a-world))
                                   (world-aliens a-world)
                                   (world-dir a-world)
                                   (world-shots a-world)))
            (define na (get-ally MY-NAME (world-allies nw)))]
           (make-package nw (list 'rckt-move (marshal-ally na))))]
        [(key=? a-key "left")
         (local
           [(define nw (make-world (move-ally-left MY-NAME
                                                   (world-allies a-world))
                                   (world-aliens a-world)
                                   (world-dir a-world)
                                   (world-shots a-world)))
            (define na (get-ally MY-NAME (world-allies nw)))]
           (make-package nw (list 'rckt-move (marshal-ally na))))]
        [(key=? a-key " ")
         (local [(define ns (process-shooting
                             (ally-rocket
                               (get-ally
                                 MY-NAME (world-allies a-world)))))
                 (define nw (make-world (world-allies a-world)
                                        (world-aliens a-world)
                                        (world-dir a-world)
                                        (cons ns (world-shots a-world))))]
           (make-package nw (list 'new-shot (marshal-shot ns))))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world key → world or package
;; Purpose: Return new package or world after a key event
(define (process-key a-world a-key)
  (cond [(key=? a-key "right")
         (local
           [(define nw (make-world (move-ally-right
                                    MY-NAME (world-allies a-world))
                                   (world-aliens a-world)
                                   (world-dir a-world)
                                   (world-shots a-world)))
            (define na (get-ally MY-NAME (world-allies nw)))]
           (make-package nw (list 'rckt-move (marshal-ally na))))]
        [(key=? a-key "left")
         (local
           [(define nw (make-world (move-ally-left MY-NAME
                                                   (world-allies a-world))
                                   (world-aliens a-world)
                                   (world-dir a-world)
                                   (world-shots a-world)))
            (define na (get-ally MY-NAME (world-allies nw)))]
           (make-package nw (list 'rckt-move (marshal-ally na))))]
        [(key=? a-key " ")
         (local [(define ns (process-shooting
                             (ally-rocket
                               (get-ally
                                 MY-NAME (world-allies a-world)))))
                 (define nw (make-world (world-allies a-world)
                                        (world-aliens a-world)
                                        (world-dir a-world)
                                        (cons ns (world-shots a-world))))]
           (make-package nw (list 'new-shot (marshal-shot ns))))]
        [else a-world]))
```

Aliens Attack Version 7

Component Implementation: Player

- Sample expressions for `process-message`

```
(define PM-RMOVE (local [(define ally (unmarshal-ally (second RM-MSG2)))]
                   (make-world (replace-ally ally (world-allies INIT-WORLD))
                               (world-aliens INIT-WORLD)
                               (world-dir INIT-WORLD)
                               (world-shots INIT-WORLD))))
```

Aliens Attack Version 7

Component Implementation: Player

- Sample expressions for `process-message`

```
(define PM-RMOVE (local [(define ally (unmarshal-ally (second RM-MSG2)))]
                   (make-world (replace-ally ally (world-allies INIT-WORLD))
                               (world-aliens INIT-WORLD)
                               (world-dir INIT-WORLD)
                               (world-shots INIT-WORLD))))
```

- ```
  (define PM-NSHOT (local [(define shot (unmarshal-shot (second NS-MSG)))]
                     (if (eq? shot NO-SHOT)
                         INIT-WORLD
                         (make-world (world-allies INIT-WORLD)
                                     (world-aliens INIT-WORLD)
                                     (world-dir INIT-WORLD)
                                     (cons shot (world-shots INIT-WORLD))))))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

- Sample expressions for process-message

```
(define PM-RMOVE (local [(define ally (unmarshal-ally (second RM-MSG2)))]
                   (make-world (replace-ally ally (world-allies INIT-WORLD))
                               (world-aliens INIT-WORLD)
                               (world-dir INIT-WORLD)
                               (world-shots INIT-WORLD))))
```

- ```
  (define PM-NSHOT (local [(define shot (unmarshal-shot (second NS-MSG)))]
                     (if (eq? shot NO-SHOT)
                         INIT-WORLD
                         (make-world (world-allies INIT-WORLD)
                                     (world-aliens INIT-WORLD)
                                     (world-dir INIT-WORLD)
                                     (cons shot (world-shots INIT-WORLD))))))
  ```

- ```
  (define PM-NSHOT2 (local [(define shot (unmarshal-shot (second NS-MSG2)))]
                      (if (eq? shot NO-SHOT)
                          INIT-WORLD
                          (make-world (world-allies INIT-WORLD)
                                      (world-aliens INIT-WORLD)
                                      (world-dir INIT-WORLD)
                                      (cons shot (world-shots INIT-WORLD))))))
  ```

Aliens Attack Version 7

Component Implementation: Player

- Sample expressions for process-message

```
(define PM-RMOVE (local [(define ally (unmarshal-ally (second RM-MSG2)))]
                   (make-world (replace-ally ally (world-allies INIT-WORLD))
                               (world-aliens INIT-WORLD)
                               (world-dir INIT-WORLD)
                               (world-shots INIT-WORLD))))
```

- ```
  (define PM-NSHOT (local [(define shot (unmarshal-shot (second NS-MSG)))]
                     (if (eq? shot NO-SHOT)
                         INIT-WORLD
                         (make-world (world-allies INIT-WORLD)
                                     (world-aliens INIT-WORLD)
                                     (world-dir INIT-WORLD)
                                     (cons shot (world-shots INIT-WORLD))))))
  ```

- ```
  (define PM-NSHOT2 (local [(define shot (unmarshal-shot (second NS-MSG2)))]
                      (if (eq? shot NO-SHOT)
                          INIT-WORLD
                          (make-world (world-allies INIT-WORLD)
                                      (world-aliens INIT-WORLD)
                                      (world-dir INIT-WORLD)
                                      (cons shot (world-shots INIT-WORLD))))))
  ```

- ```
  (define PM-NALLY (local [(define ally (unmarshal-ally (second NA-MSG)))]
                     (make-world (cons ally (world-allies INIT-WORLD))
                                 (world-aliens INIT-WORLD)
                                 (world-dir INIT-WORLD)
                                 (world-shots INIT-WORLD))))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

- Sample expressions for process-message

```
(define PM-RMOVE (local [(define ally (unmarshal-ally (second RM-MSG2)))]
                   (make-world (replace-ally ally (world-allies INIT-WORLD))
                               (world-aliens INIT-WORLD)
                               (world-dir INIT-WORLD)
                               (world-shots INIT-WORLD))))
```

- ```
(define PM-NSHOT (local [(define shot (unmarshal-shot (second NS-MSG)))]
                   (if (eq? shot NO-SHOT)
                       INIT-WORLD
                       (make-world (world-allies INIT-WORLD)
                                   (world-aliens INIT-WORLD)
                                   (world-dir INIT-WORLD)
                                   (cons shot (world-shots INIT-WORLD))))))
```

- ```
(define PM-NSHOT2 (local [(define shot (unmarshal-shot (second NS-MSG2)))]
                   (if (eq? shot NO-SHOT)
                       INIT-WORLD
                       (make-world (world-allies INIT-WORLD)
                                   (world-aliens INIT-WORLD)
                                   (world-dir INIT-WORLD)
                                   (cons shot (world-shots INIT-WORLD))))))
```

- ```
(define PM-NALLY (local [(define ally (unmarshal-ally (second NA-MSG)))]
                   (make-world (cons ally (world-allies INIT-WORLD))
                               (world-aliens INIT-WORLD)
                               (world-dir INIT-WORLD)
                               (world-shots INIT-WORLD))))
```

- ```
(define PM-RMALLY (local [(define name (second RA-MSG2))]
                    (make-world (remove-ally name (world-allies INIT-WORLD2))
                                (world-aliens INIT-WORLD2)
                                (world-dir INIT-WORLD2)
                                (world-shots INIT-WORLD2))))
```

Aliens Attack Version 7

Component Implementation: Player

- ```
  (define PM-SWORLD
          (local [(define name (second SW-MSG2))]
            (make-package WORLD3 (cons 'world-back
                                       (cons name (marshal-world WORLD3))))))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

- ```
  (define PM-SWORLD
          (local [(define name (second SW-MSG2))]
            (make-package WORLD3 (cons 'world-back
                                       (cons name (marshal-world WORLD3))))))
  ```
- ```
  (define PM-START (local [(define world  (unmarshal-world (rest ST-MSG)))
                           (define allies (world-allies world))
                           (define aliens (world-aliens world))
                           (define dir    (world-dir world))
                           (define shots  (world-shots world))]
                     (make-world (cons INIT-ALLY allies) aliens dir shots)))
  ```

Aliens Attack Version 7

Component Implementation: Player

- ```
  ;; world tpm → world or package  Purpose: Process given to-player message
  (define (process-message  a-world a-tpm)
    (local [(define tag (first a-tpm))]
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'new-shot)
           (local [(define shot (unmarshal-shot (second a-tpm)))]
             (if (eq? shot NO-SHOT) a-world
                 (make-world (world-allies a-world) (world-aliens a-world)
                   (world-dir a-world) (cons shot (world-shots a-world)))))])))
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'new-shot)
           (local [(define shot (unmarshal-shot (second a-tpm)))]
             (if (eq? shot NO-SHOT) a-world
                 (make-world (world-allies a-world) (world-aliens a-world)
                   (world-dir a-world) (cons shot (world-shots a-world)))))]
          [(eq? tag 'new-ally)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (cons ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'new-shot)
           (local [(define shot (unmarshal-shot (second a-tpm)))]
             (if (eq? shot NO-SHOT) a-world
                 (make-world (world-allies a-world) (world-aliens a-world)
                   (world-dir a-world) (cons shot (world-shots a-world)))))]
          [(eq? tag 'new-ally)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (cons ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'rm-ally)
           (local [(define name (second a-tpm))]
             (make-world (remove-ally name (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'new-shot)
           (local [(define shot (unmarshal-shot (second a-tpm)))]
             (if (eq? shot NO-SHOT) a-world
                 (make-world (world-allies a-world) (world-aliens a-world)
                   (world-dir a-world) (cons shot (world-shots a-world)))))]
          [(eq? tag 'new-ally)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (cons ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'rm-ally)
           (local [(define name (second a-tpm))]
             (make-world (remove-ally name (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'send-world)
           (local [(define name (second a-tpm))]
             (make-package a-world
                           (cons 'world-back (cons name (marshal-world a-world)))))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'new-shot)
           (local [(define shot (unmarshal-shot (second a-tpm)))]
             (if (eq? shot NO-SHOT) a-world
                 (make-world (world-allies a-world) (world-aliens a-world)
                   (world-dir a-world) (cons shot (world-shots a-world)))))]
          [(eq? tag 'new-ally)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (cons ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'rm-ally)
           (local [(define name (second a-tpm))]
             (make-world (remove-ally name (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'send-world)
           (local [(define name (second a-tpm))]
             (make-package a-world
                           (cons 'world-back (cons name (marshal-world a-world)))))]
          [(eq? tag 'start)
           (local
            [(define world (unmarshal-world (rest a-tpm)))
             (define allies (world-allies world)) (define aliens (world-aliens world))
             (define dir (world-dir world))
             (define shots  (world-shots world))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'new-shot)
           (local [(define shot (unmarshal-shot (second a-tpm)))]
             (if (eq? shot NO-SHOT) a-world
                 (make-world (world-allies a-world) (world-aliens a-world)
                   (world-dir a-world) (cons shot (world-shots a-world)))))]
          [(eq? tag 'new-ally)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (cons ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'rm-ally)
           (local [(define name (second a-tpm))]
             (make-world (remove-ally name (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'send-world)
           (local [(define name (second a-tpm))]
             (make-package a-world
                           (cons 'world-back (cons name (marshal-world a-world)))))]
          [(eq? tag 'start)
           (local
            [(define world (unmarshal-world (rest a-tpm)))
             (define allies (world-allies world)) (define aliens (world-aliens world))
             (define dir (world-dir world))
             (define shots  (world-shots world))]
            (make-world (cons INIT-ALLY allies) aliens dir shots))]
```

Aliens Attack Version 7

Component Implementation: Player

```
;; world tpm → world or package  Purpose: Process given to-player message
(define (process-message  a-world a-tpm)
  (local [(define tag (first a-tpm))]
    (cond [(eq? tag 'rckt-move)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (replace-ally ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'new-shot)
           (local [(define shot (unmarshal-shot (second a-tpm)))]
             (if (eq? shot NO-SHOT) a-world
                 (make-world (world-allies a-world) (world-aliens a-world)
                   (world-dir a-world) (cons shot (world-shots a-world)))))]
          [(eq? tag 'new-ally)
           (local [(define ally (unmarshal-ally (second a-tpm)))]
             (make-world (cons ally (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'rm-ally)
           (local [(define name (second a-tpm))]
             (make-world (remove-ally name (world-allies a-world))
               (world-aliens a-world) (world-dir a-world) (world-shots a-world)))]
          [(eq? tag 'send-world)
           (local [(define name (second a-tpm))]
             (make-package a-world
                           (cons 'world-back (cons name (marshal-world a-world)))))]
          [(eq? tag 'start)
           (local
            [(define world (unmarshal-world (rest a-tpm)))
             (define allies (world-allies world)) (define aliens (world-aliens world))
             (define dir (world-dir world))
             (define shots  (world-shots world))]
            (make-world (cons INIT-ALLY allies) aliens dir shots))]
          [else (error (format "Unknown message type received: ~s" (first a-tpm)))])))
```

Aliens Attack Version 7

Component Implementation: Player

- ```
  ;; Tests using sample computations for process-message
  (check-expect (process-message INIT-WORLD  RM-MSG2) PM-RMOVE)
  (check-expect (process-message INIT-WORLD  NS-MSG)  PM-NSHOT)
  (check-expect (process-message INIT-WORLD  NA-MSG)  PM-NALLY)
  (check-expect (process-message INIT-WORLD2 RA-MSG2) PM-RMALLY)
  (check-expect (process-message WORLD3      SW-MSG2) PM-SWORLD)
  (check-expect (process-message INIT-WORLD  ST-MSG)  PM-START)

  ;; Tests using sample computations for process-message
  (check-expect
    (process-message (make-world (list (make-ally 6 "Doris"))
                                 (list (make-posn 4 9))
                                 "right"
                                 '())
                                 (list 'send-world "Don Marco"))
    (make-package (make-world (list (make-ally 6 "Doris"))
                              (list (make-posn 4 9))
                              "right"
                              '())
                  (cons
                    'world-back
                    (cons "Don Marco"
                          (list (list (list 6 "Doris"))
                                (list (list 4 9))
                                "right"
                                '())))))

  (check-error
    (process-message INIT-WORLD (list 'send-w "Magna"))
    "Unknown message type received: send-w")
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by `process-message` are straight-forward list-processing exercises

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by `process-message` are straight-forward list-processing exercises

- ```
  ;; Sample expressions for remove-ally
  (define RM-ALLIES1 (filter (λ (a) (not (string=? (ally-name a) "Quintana")))
                             INIT-ALLIES))
  (define RM-ALLIES2 (filter (λ (a) (not (string=? (ally-name a) "Marco")))
                             ALLIES2))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by `process-message` are straight-forward list-processing exercises
- ```
  ;; string lor → lor
  ;; Purpose: Remove given ally for given list of allies
  (define (remove-ally a-name a-lor)
  ```
- ```
  ;; Sample expressions for remove-ally
  (define RM-ALLIES1 (filter (λ (a) (not (string=? (ally-name a) "Quintana")))
                             INIT-ALLIES))
  (define RM-ALLIES2 (filter (λ (a) (not (string=? (ally-name a) "Marco")))
                             ALLIES2))
  ```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by process-message are straight-forward list-processing exercises
- ```
  ;; string lor → lor
  ;; Purpose: Remove given ally for given list of allies
  (define (remove-ally a-name a-lor)
  ```
- ```
  ;; Sample expressions for remove-ally
  (define RM-ALLIES1 (filter (λ (a) (not (string=? (ally-name a) "Quintana")))
                             INIT-ALLIES))
  (define RM-ALLIES2 (filter (λ (a) (not (string=? (ally-name a) "Marco")))
                             ALLIES2))
  ```
- ```
  ;; Tests using sample computations for remove-ally
  (check-expect (remove-ally "Quintana" INIT-ALLIES) RM-ALLIES1)
  (check-expect (remove-ally "Marco"    ALLIES2)     RM-ALLIES2)

  ;; Tests using sample values for remove-ally
  (check-expect
   (remove-ally
     "Driscoll"
     (list (make-ally 4 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))
   (list (make-ally 4 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))

  (check-expect
   (remove-ally
    "Driscoll"
    (list (make-ally 12 "Sakas")    (make-ally 14 "Davila")
          (make-ally 11 "Driscoll") (make-ally 17 "Morazan")))
    (list (make-ally 12 "Sakas") (make-ally 14 "Davila") (make-ally 17 "Morazan")))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by process-message are straight-forward list-processing exercises
- ```
  ;; string lor → lor
  ;; Purpose: Remove given ally for given list of allies
  (define (remove-ally a-name a-lor)
  ```
- ```
    (filter (λ (a) (not (string=? (ally-name a) a-name))) a-lor))
  ```
- ```
  ;; Sample expressions for remove-ally
  (define RM-ALLIES1 (filter (λ (a) (not (string=? (ally-name a) "Quintana")))
                             INIT-ALLIES))
  (define RM-ALLIES2 (filter (λ (a) (not (string=? (ally-name a) "Marco")))
                             ALLIES2))
  ```
- ```
  ;; Tests using sample computations for remove-ally
  (check-expect (remove-ally "Quintana" INIT-ALLIES) RM-ALLIES1)
  (check-expect (remove-ally "Marco"    ALLIES2)     RM-ALLIES2)

  ;; Tests using sample values for remove-ally
  (check-expect
   (remove-ally
     "Driscoll"
     (list (make-ally 4 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))
   (list (make-ally 4 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))

  (check-expect
   (remove-ally
    "Driscoll"
    (list (make-ally 12 "Sakas")    (make-ally 14 "Davila")
          (make-ally 11 "Driscoll") (make-ally 17 "Morazan")))
    (list (make-ally 12 "Sakas") (make-ally 14 "Davila") (make-ally 17 "Morazan")))
  ```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by `process-message` are straight-forward list-processing exercises

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by `process-message` are straight-forward list-processing exercises

-
```
;; Sample expressions for replace-ally
(define RP-ALLIES1 (map (λ (a)
                          (if (string=? (ally-name a) (ally-name INIT-ALLY2))
                              INIT-ALLY2
                              a))
                        INIT-ALLIES))
(define RP-ALLIES2 (map (λ (a)
                          (if (string=? (ally-name a) (ally-name INIT-ALLY))
                              INIT-ALLY
                              a))
                        ALLIES2))
```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by `process-message` are straight-forward list-processing exercises
-
```
;; ally lor → lor
;; Purpose: Replace given ally in given lor
(define (replace-ally an-ally a-lor)
```

-
```
;; Sample expressions for replace-ally
(define RP-ALLIES1 (map (λ (a)
                          (if (string=? (ally-name a) (ally-name INIT-ALLY2))
                              INIT-ALLY2
                              a))
                        INIT-ALLIES))
(define RP-ALLIES2 (map (λ (a)
                          (if (string=? (ally-name a) (ally-name INIT-ALLY))
                              INIT-ALLY
                              a))
                        ALLIES2))
```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by process-message are straight-forward list-processing exercises
- ```
  ;; ally lor → lor
  ;; Purpose: Replace given ally in given lor
  (define (replace-ally an-ally a-lor)
  ```

- ```
  ;; Sample expressions for replace-ally
  (define RP-ALLIES1 (map (λ (a)
                            (if (string=? (ally-name a) (ally-name INIT-ALLY2))
                                INIT-ALLY2
                                a))
                          INIT-ALLIES))
  (define RP-ALLIES2 (map (λ (a)
                            (if (string=? (ally-name a) (ally-name INIT-ALLY))
                                INIT-ALLY
                                a))
                          ALLIES2))
  ```

- ```
  ;; Tests using sample computations for replace-ally
  (check-expect (replace-ally INIT-ALLY2 INIT-ALLIES) RP-ALLIES1)
  (check-expect (replace-ally INIT-ALLY  ALLIES2)     RP-ALLIES2)

  ;; Tests using sample values for replace-ally
  (check-expect
    (replace-ally
      (make-ally 5 "Manfred")
      (list (make-ally 4 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))
    (list (make-ally 5 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))

  (check-expect
  ```
  ```

Aliens Attack Version 7

Component Implementation: Player

- The auxiliary functions needed by process-message are straight-forward list-processing exercises
- ```
  ;; ally lor → lor
  ;; Purpose: Replace given ally in given lor
  (define (replace-ally an-ally a-lor)
  ```
- ```
    (map (λ (a) (if (string=? (ally-name a) (ally-name an-ally))
                    an-ally
                    a))
         a-lor))
  ```
- ```
  ;; Sample expressions for replace-ally
  (define RP-ALLIES1 (map (λ (a)
                            (if (string=? (ally-name a) (ally-name INIT-ALLY2))
                                INIT-ALLY2
                                a))
                          INIT-ALLIES))
  (define RP-ALLIES2 (map (λ (a)
                            (if (string=? (ally-name a) (ally-name INIT-ALLY))
                                INIT-ALLY
                                a))
                          ALLIES2))
  ```
- ```
  ;; Tests using sample computations for replace-ally
  (check-expect (replace-ally INIT-ALLY2 INIT-ALLIES) RP-ALLIES1)
  (check-expect (replace-ally INIT-ALLY  ALLIES2)     RP-ALLIES2)

  ;; Tests using sample values for replace-ally
  (check-expect
    (replace-ally
      (make-ally 5 "Manfred")
      (list (make-ally 4 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))
    (list (make-ally 5 "Manfred") (make-ally 7 "Cordula") (make-ally 1 "Catherina")))

  (check-expect
  ```

Aliens Attack Version 7

Component Implementation: Player

- The updated `run` function for the player is:

```
;; string → world
;; Purpose: To run the game
(define (run a-name)
  (local [(define TICK-RATE 1/4)]
    (big-bang
      INIT-WORLD
      [on-draw draw-world]
      [name MY-NAME]
      [on-key process-key]
      [on-tick process-tick TICK-RATE]
      [stop-when game-over? draw-last-world]
      [register LOCALHOST]
      [on-receive process-message])))
```

Aliens Attack Version 7

Component Implementation: Server

- Server requires 3 handlers: process messages players, manage new players, and manage departure of players
- Server must track players

Aliens Attack Version 7

Component Implementation: Server

- Server requires 3 handlers: process messages players, manage new players, and manage departure of players
- Server must track players
- Based on this we may define the `universe` and the `run-server` function as follows:

```
;; A universe is a (listof iworld), where each iworld has
;; a unique name

;; Sample instances of universe
(define INIT-UNIV '())
(define A-UNIV    (list iworld1 iworld2))

;; Z → universe
;; Purpose: Run the chat server
(define (run-server a-z)
  (universe INIT-UNIV
            (on-new        add-player)
            (on-msg        process-message)
            (on-disconnect rm-player)))
```

Aliens Attack Version 7

Component Implementation: Server

- Player not already used in the `universe` it is allowed to join
- Otherwise it is rejected

Aliens Attack Version 7

Component Implementation: Server

- Player not already used in the `universe` it is allowed to join
- Otherwise it is rejected
- There are two cases according to our protocol design

Aliens Attack Version 7

Component Implementation: Server

- Player not already used in the `universe` it is allowed to join
- Otherwise it is rejected
- There are two cases according to our protocol design
- First player to join then the initial `world` must be sent to it
- Not the first player to join then the existing players must be sent a new ally message and that the first `world` in the universe must be sent a send world message

Aliens Attack Version 7

Component Implementation: Server

- Player not already used in the `universe` it is allowed to join
- Otherwise it is rejected
- There are two cases according to our protocol design
- First player to join then the initial `world` must be sent to it
- Not the first player to join then the existing players must be sent a new ally message and that the first `world` in the universe must be sent a send world message
- Sample expressions:

```
(define RPT-ADD (make-bundle A-UNIV '() (list iworld1)))
```

Aliens Attack Version 7

Component Implementation: Server

- Player not already used in the `universe` it is allowed to join
- Otherwise it is rejected
- There are two cases according to our protocol design
- First player to join then the initial `world` must be sent to it
- Not the first player to join then the existing players must be sent a new ally message and that the first `world` in the universe must be sent a send world message
- Sample expressions:

```
(define RPT-ADD (make-bundle A-UNIV '() (list iworld1)))
```

- ```
  (define EMP-ADD (make-bundle
                   (cons iworld1 INIT-UNIV)
                   (list (make-mail
                          iworld1
                          (cons 'start (marshal-world INIT-WORLD))))
                   '()))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Server

- Player not already used in the `universe` it is allowed to join
- Otherwise it is rejected
- There are two cases according to our protocol design
- First player to join then the initial `world` must be sent to it
- Not the first player to join then the existing players must be sent a new ally message and that the first `world` in the universe must be sent a send world message
- Sample expressions:

```
(define RPT-ADD (make-bundle A-UNIV '() (list iworld1)))
```

- ```
  (define EMP-ADD (make-bundle
                   (cons iworld1 INIT-UNIV)
                   (list (make-mail
                          iworld1
                          (cons 'start (marshal-world INIT-WORLD))))
                   '()))
  ```

- ```
  (define NEW-ADD (make-bundle
                   (cons iworld3 A-UNIV)
                   (cons (make-mail
                          (first A-UNIV)
                          (list 'send-world (iworld-name iworld3)))
                         (map (λ (iw)
                                (make-mail
                                 iw
                                 (list 'new-ally (list INIT-ROCKET iworld-name iworld3)))))
                              A-UNIV))
                   '()))
  ```

Aliens Attack Version 7

Component Implementation: Server

- `;; universe iworld → bundle  Purpose: Add new world to the universe`
  `(define (add-player a-univ an-iw)`

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld → bundle  Purpose: Add new world to the universe
  (define (add-player a-univ an-iw)
  ```

- ```
  ;; Tests using sample computations for add-player
  (check-expect (add-player A-UNIV    iworld1) RPT-ADD)
  (check-expect (add-player INIT-UNIV iworld1) EMP-ADD)
  (check-expect (add-player A-UNIV    iworld3) NEW-ADD)
  ;; Tests using sample values for add-player
  (check-expect
    (add-player (list iworld2 iworld3) iworld1)
    (make-bundle
      (list iworld1 iworld2 iworld3)
      (cons (make-mail
              (first (list iworld2 iworld3))
              (list 'send-world "iworld1"))
            (map (λ (iw) (make-mail iw (list 'new-ally (list INIT-ROCKET "iworld1"))))
                 (list iworld2 iworld3)))
      '()))
  ```

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld → bundle  Purpose: Add new world to the universe
  (define (add-player a-univ an-iw)
  ```
- ```
    (cond
      [(member? (iworld-name an-iw) (map iworld-name a-univ))
       (make-bundle a-univ '() (list an-iw))]
  ```

- ```
  ;; Tests using sample computations for add-player
  (check-expect (add-player A-UNIV    iworld1) RPT-ADD)
  (check-expect (add-player INIT-UNIV iworld1) EMP-ADD)
  (check-expect (add-player A-UNIV    iworld3) NEW-ADD)
  ;; Tests using sample values for add-player
  (check-expect
    (add-player (list iworld2 iworld3) iworld1)
    (make-bundle
      (list iworld1 iworld2 iworld3)
      (cons (make-mail
              (first (list iworld2 iworld3))
              (list 'send-world "iworld1"))
            (map (λ (iw) (make-mail iw (list 'new-ally (list INIT-ROCKET "iworld1"))))
                 (list iworld2 iworld3)))
      '()))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld → bundle  Purpose: Add new world to the universe
  (define (add-player a-univ an-iw)
  ```
- ```
    (cond
      [(member? (iworld-name an-iw) (map iworld-name a-univ))
       (make-bundle a-univ '() (list an-iw))]
  ```
- ```
      [(empty? a-univ)
       (make-bundle (cons an-iw a-univ)
                    (list (make-mail an-iw (cons 'start (marshal-world INIT-WORLD))))
                    '())]
  ```

- ```
  ;; Tests using sample computations for add-player
  (check-expect (add-player A-UNIV    iworld1) RPT-ADD)
  (check-expect (add-player INIT-UNIV iworld1) EMP-ADD)
  (check-expect (add-player A-UNIV    iworld3) NEW-ADD)
  ;; Tests using sample values for add-player
  (check-expect
    (add-player (list iworld2 iworld3) iworld1)
    (make-bundle
      (list iworld1 iworld2 iworld3)
      (cons (make-mail
              (first (list iworld2 iworld3))
              (list 'send-world "iworld1"))
            (map (λ (iw) (make-mail iw (list 'new-ally (list INIT-ROCKET "iworld1"))))
                 (list iworld2 iworld3)))
      '()))
  ```

Aliens Attack Version 7

Component Implementation: Server

```
;; universe iworld → bundle  Purpose: Add new world to the universe
(define (add-player a-univ an-iw)
  (cond
    [(member? (iworld-name an-iw) (map iworld-name a-univ))
     (make-bundle a-univ '() (list an-iw))]
    [(empty? a-univ)
     (make-bundle (cons an-iw a-univ)
                  (list (make-mail an-iw (cons 'start (marshal-world INIT-WORLD))))
                  '())]
    [else (make-bundle
           (cons an-iw a-univ)
           (cons (make-mail (first a-univ) (list 'send-world (iworld-name an-iw)))
           (map (λ (iw)
                  (make-mail iw (list 'new-ally (list INIT-ROCKET (iworld-name an-iw)))))
                a-univ))
           '())]))
;; Tests using sample computations for add-player
(check-expect (add-player A-UNIV    iworld1) RPT-ADD)
(check-expect (add-player INIT-UNIV iworld1) EMP-ADD)
(check-expect (add-player A-UNIV    iworld3) NEW-ADD)
;; Tests using sample values for add-player
(check-expect
  (add-player (list iworld2 iworld3) iworld1)
  (make-bundle
    (list iworld1 iworld2 iworld3)
    (cons (make-mail
           (first (list iworld2 iworld3))
           (list 'send-world "iworld1"))
          (map (λ (iw) (make-mail iw (list 'new-ally (list INIT-ROCKET "iworld1"))))
               (list iworld2 iworld3)))
    '()))
```

Aliens Attack Version 7

Component Implementation: Server

- Removing a player that disconnects from the universe: new `universe` and messages to the remaining players

Aliens Attack Version 7

Component Implementation: Server

- Removing a player that disconnects from the universe: new `universe` and messages to the remaining players
- Sample expression:

```
(define IW1-RM
  (local
    [(define new-univ
       (filter
         (λ (iw)
           (not (string=? (iworld-name iw) (iworld-name iworld1))))
         A-UNIV))]
    (make-bundle
      new-univ
      (map (λ (iw)
             (make-mail iw
                        (list 'rm-ally (iworld-name iworld1))))
           new-univ)
      '())))
```

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld → bundle
  ;; Purpose: Remove a player from the game
  (define (rm-player a-univ an-iw)
  ```
  ```

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld → bundle
  ;; Purpose: Remove a player from the game
  (define (rm-player a-univ an-iw)
  ```

- ```
  ;; Tests using sample computations for rm-player
  (check-expect (rm-player A-UNIV iworld1) IW1-RM)

  ;; Tests using sample values for rm-player
  (check-expect
    (rm-player (list iworld1 iworld2 iworld3) iworld2)
    (make-bundle (list iworld1 iworld3)
                 (map
                   (λ (iw)
                     (make-mail iw (list 'rm-ally "iworld2")))
                   (list iworld1 iworld3))
                 '()))
  ```

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld → bundle
  ;; Purpose: Remove a player from the game
  (define (rm-player a-univ an-iw)
  ```
- ```
    (local
      [(define new-univ (filter
                         (λ (iw)
                           (not (string=? (iworld-name iw) (iworld-name an-iw))))
                         a-univ))]
      (make-bundle
       new-univ
       (map (λ (iw)
              (make-mail iw (list 'rm-ally (iworld-name an-iw))))
            new-univ)
       '())))
  ```
- ```
  ;; Tests using sample computations for rm-player
  (check-expect (rm-player A-UNIV iworld1) IW1-RM)

  ;; Tests using sample values for rm-player
  (check-expect
    (rm-player (list iworld1 iworld2 iworld3) iworld2)
    (make-bundle (list iworld1 iworld3)
                 (map
                   (λ (iw)
                     (make-mail iw (list 'rm-ally "iworld2")))
                   (list iworld1 iworld3))
                 '()))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Server

- `process-message` processes all incoming `tsms`
- According to the protocol diagrams the server always sends out one or more messages when it receives a `tsm`
- This means the this handler must return a `bundle`

Aliens Attack Version 7

Component Implementation: Server

- Sample expressions:

```
(define PM-RM
  (local [(define tag (first SRM-MSG))
          (define send-list
                  (if (eq? tag 'world-back)
                      (filter (λ (iw) (string=? (iworld-name iw) (second SRM-MSG)))
                              A-UNIV)
                      (filter (λ (iw)
                                (not (string=? (iworld-name iw) (iworld-name iworld1))))
                              A-UNIV)))]
    (make-bundle A-UNIV (map (λ (iw) (make-mail iw SRM-MSG)) send-list) '())))
```

Aliens Attack Version 7

Component Implementation: Server

- Sample expressions:

```
(define PM-RM
  (local [(define tag (first SRM-MSG))
          (define send-list
                  (if (eq? tag 'world-back)
                      (filter (λ (iw) (string=? (iworld-name iw) (second SRM-MSG)))
                              A-UNIV)
                      (filter (λ (iw)
                                 (not (string=? (iworld-name iw) (iworld-name iworld1))))
                              A-UNIV)))]
    (make-bundle A-UNIV (map (λ (iw) (make-mail iw SRM-MSG)) send-list) '())))
```

-

```
(define PM-NS
  (local [(define tag (first SNS-MSG))
          (define send-list
           (if (eq? tag 'world-back)
               (filter (λ (iw) (string=? (iworld-name iw) (second SNS-MSG)))
                       A-UNIV)
               (filter (λ (iw) (not (string=? (iworld-name iw) iworld-name iworld1))))
                       A-UNIV)))]
    (make-bundle A-UNIV (map (λ (iw) (make-mail iw SNS-MSG)) send-list) '())))
```

Aliens Attack Version 7

Component Implementation: Server

- Sample expressions:

```
(define PM-RM
  (local [(define tag (first SRM-MSG))
          (define send-list
                  (if (eq? tag 'world-back)
                      (filter (λ (iw) (string=? (iworld-name iw) (second SRM-MSG)))
                              A-UNIV)
                      (filter (λ (iw)
                                 (not (string=? (iworld-name iw) (iworld-name iworld1))))
                              A-UNIV)))]
    (make-bundle A-UNIV (map (λ (iw) (make-mail iw SRM-MSG)) send-list) '())))
```

-

```
(define PM-NS
  (local [(define tag (first SNS-MSG))
          (define send-list
           (if (eq? tag 'world-back)
               (filter (λ (iw) (string=? (iworld-name iw) (second SNS-MSG)))
                       A-UNIV)
               (filter (λ (iw) (not (string=? (iworld-name iw) iworld-name iworld1))))
                       A-UNIV)))]
    (make-bundle A-UNIV (map (λ (iw) (make-mail iw SNS-MSG)) send-list) '())))
```

-

```
(define PM-WB
  (local [(define tag (first SWB-MSG))
          (define send-list
           (if (eq? tag 'world-back)
               (filter (λ (iw) (string=? (iworld-name iw) (second SWB-MSG))) A-UNIV)
               (filter (λ (iw) (not (string=? (iworld-name iw) (iworld-name iworld1))))
                       A-UNIV)))]
    (make-bundle A-UNIV
                 (map (λ (iw)
                         (make-mail (first send-list) (cons 'start (rest (rest SWB-MSG)))))
                      send-list)
                 '())))
```

Aliens Attack Version 7

Component Implementation: Server

- `;; universe iworld tsm → bundle  Purpose: Process given message from given world`
  `(define (process-message a-univ an-iw a-tsm)`

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld tsm → bundle  Purpose: Process given message from given world
  (define (process-message a-univ an-iw a-tsm)
  ```

- ```
  ;; Tests using sample computations for process-message
  (check-expect (process-message A-UNIV iworld1 SRM-MSG) PM-RM)
  (check-expect (process-message A-UNIV iworld1 SNS-MSG) PM-NS)
  (check-expect (process-message A-UNIV iworld2 SWB-MSG) PM-WB)

  ;; Tests using sample values for process-message
  (check-error
    (process-message A-UNIV iworld2 '(rocket-move ((5 "iworld2"))))
    (format "Unknown message received by server: ~s." '(rocket-move ((5 "iworld2")))))
  ```

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld tsm → bundle  Purpose: Process given message from given world
  (define (process-message a-univ an-iw a-tsm)
  ```
- ```
    (local
      [(define tag (first a-tsm))
       (define send-list
         (if (eq? tag 'world-back)
             (filter (λ (iw) (string=? (iworld-name iw) (second a-tsm))) a-univ)
             (filter (λ (iw) (not (string=? (iworld-name iw) (iworld-name an-iw))))
                     a-univ)))]
  ```

- ```
  ;; Tests using sample computations for process-message
  (check-expect (process-message A-UNIV iworld1 SRM-MSG) PM-RM)
  (check-expect (process-message A-UNIV iworld1 SNS-MSG) PM-NS)
  (check-expect (process-message A-UNIV iworld2 SWB-MSG) PM-WB)

  ;; Tests using sample values for process-message
  (check-error
    (process-message A-UNIV iworld2 '(rocket-move ((5 "iworld2"))))
    (format "Unknown message received by server: ~s." '(rocket-move ((5 "iworld2")))))
  ```
  ```

Aliens Attack Version 7

Component Implementation: Server

- ```
  ;; universe iworld tsm → bundle  Purpose: Process given message from given world
  (define (process-message a-univ an-iw a-tsm)
  ```
- ```
    (local
      [(define tag (first a-tsm))
       (define send-list
         (if (eq? tag 'world-back)
             (filter (λ (iw) (string=? (iworld-name iw) (second a-tsm))) a-univ)
             (filter (λ (iw) (not (string=? (iworld-name iw) (iworld-name an-iw))))
                     a-univ)))]
  ```
- ```
      (cond [(eq? tag 'rckt-move)
             (make-bundle a-univ (map (λ (iw) (make-mail iw a-tsm)) send-list) '())]
  ```

- ```
  ;; Tests using sample computations for process-message
  (check-expect (process-message A-UNIV iworld1 SRM-MSG) PM-RM)
  (check-expect (process-message A-UNIV iworld1 SNS-MSG) PM-NS)
  (check-expect (process-message A-UNIV iworld2 SWB-MSG) PM-WB)

  ;; Tests using sample values for process-message
  (check-error
    (process-message A-UNIV iworld2 '(rocket-move ((5 "iworld2"))))
    (format "Unknown message received by server: ~s." '(rocket-move ((5 "iworld2")))))
  ```

Aliens Attack Version 7

Component Implementation: Server

```
;; universe iworld tsm → bundle  Purpose: Process given message from given world
(define (process-message a-univ an-iw a-tsm)
  (local
    [(define tag (first a-tsm))
     (define send-list
       (if (eq? tag 'world-back)
           (filter (λ (iw) (string=? (iworld-name iw) (second a-tsm))) a-univ)
           (filter (λ (iw) (not (string=? (iworld-name iw) (iworld-name an-iw))))
                   a-univ)))]
    (cond [(eq? tag 'rckt-move)
           (make-bundle a-univ (map (λ (iw) (make-mail iw a-tsm)) send-list) '())]
          [(eq? tag 'new-shot)
           (make-bundle a-univ (map (λ (iw) (make-mail iw a-tsm)) send-list) '())]
```

```
;; Tests using sample computations for process-message
(check-expect (process-message A-UNIV iworld1 SRM-MSG) PM-RM)
(check-expect (process-message A-UNIV iworld1 SNS-MSG) PM-NS)
(check-expect (process-message A-UNIV iworld2 SWB-MSG) PM-WB)

;; Tests using sample values for process-message
(check-error
  (process-message A-UNIV iworld2 '(rocket-move ((5 "iworld2"))))
  (format "Unknown message received by server: ~s." '(rocket-move ((5 "iworld2"))))))
```

Aliens Attack Version 7

Component Implementation: Server

```
;; universe iworld tsm → bundle  Purpose: Process given message from given world
(define (process-message a-univ an-iw a-tsm)
  (local
    [(define tag (first a-tsm))
     (define send-list
       (if (eq? tag 'world-back)
           (filter (λ (iw) (string=? (iworld-name iw) (second a-tsm))) a-univ)
           (filter (λ (iw) (not (string=? (iworld-name iw) (iworld-name an-iw))))
                   a-univ)))]
    (cond [(eq? tag 'rckt-move)
           (make-bundle a-univ (map (λ (iw) (make-mail iw a-tsm)) send-list) '())]
          [(eq? tag 'new-shot)
           (make-bundle a-univ (map (λ (iw) (make-mail iw a-tsm)) send-list) '())]
          [(eq? tag 'world-back)
           (make-bundle a-univ
                        (list (make-mail (first send-list)
                                         (cons 'start (rest (rest a-tsm)))))
                        '())]
```

```
;; Tests using sample computations for process-message
(check-expect (process-message A-UNIV iworld1 SRM-MSG) PM-RM)
(check-expect (process-message A-UNIV iworld1 SNS-MSG) PM-NS)
(check-expect (process-message A-UNIV iworld2 SWB-MSG) PM-WB)

;; Tests using sample values for process-message
(check-error
  (process-message A-UNIV iworld2 '(rocket-move ((5 "iworld2"))))
  (format "Unknown message received by server: ~s." '(rocket-move ((5 "iworld2"))))
```

Aliens Attack Version 7

Component Implementation: Server

```
;; universe iworld tsm → bundle  Purpose: Process given message from given world
(define (process-message a-univ an-iw a-tsm)
  (local
    [(define tag (first a-tsm))
     (define send-list
       (if (eq? tag 'world-back)
           (filter (λ (iw) (string=? (iworld-name iw) (second a-tsm))) a-univ)
           (filter (λ (iw) (not (string=? (iworld-name iw) (iworld-name an-iw))))
                   a-univ)))]
    (cond [(eq? tag 'rckt-move)
           (make-bundle a-univ (map (λ (iw) (make-mail iw a-tsm)) send-list) '())]
          [(eq? tag 'new-shot)
           (make-bundle a-univ (map (λ (iw) (make-mail iw a-tsm)) send-list) '())]
          [(eq? tag 'world-back)
           (make-bundle a-univ
                        (list (make-mail (first send-list)
                                         (cons 'start (rest (rest a-tsm)))))
                        '())]
          [else
           (error (format "Unknown message received by server: ~s."
                          a-tsm))])))
;; Tests using sample computations for process-message
(check-expect (process-message A-UNIV iworld1 SRM-MSG) PM-RM)
(check-expect (process-message A-UNIV iworld1 SNS-MSG) PM-NS)
(check-expect (process-message A-UNIV iworld2 SWB-MSG) PM-WB)

;; Tests using sample values for process-message
(check-error
  (process-message A-UNIV iworld2 '(rocket-move ((5 "iworld2"))))
  (format "Unknown message received by server: ~s." '(rocket-move ((5 "iworld2")))))
```

Aliens Attack Version 7

A Subtle Bug

- Make sure you have at least two copies of the player's program saved (each with a unique `MY-NAME` value)
- To play the game as multiple players on your machine remember to first call `run-server` in the server file and then `run` for each of the player files.

Aliens Attack Version 7

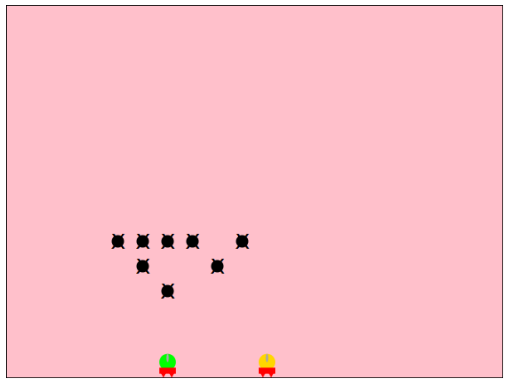
A Subtle Bug

- Make sure you have at least two copies of the player's program saved (each with a unique `MY-NAME` value)
- To play the game as multiple players on your machine remember to first call `run-server` in the server file and then `run` for each of the player files.
- What do you notice, if anything, after playing the game a few times on your computer?

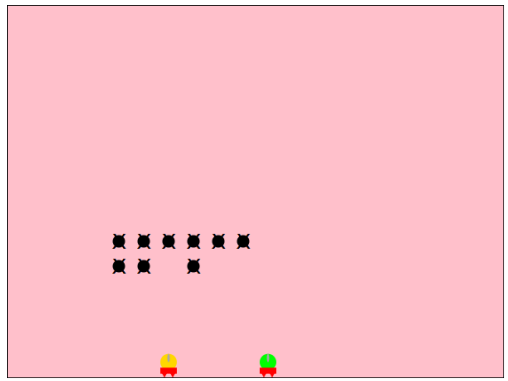
Aliens Attack Version 7

A Subtle Bug

- Make sure you have at least two copies of the player's program saved (each with a unique `MY-NAME` value)
- To play the game as multiple players on your machine remember to first call `run-server` in the server file and then `run` for each of the player files.
- What do you notice, if anything, after playing the game a few times on your computer?
- If you run it enough times you are bound to see that there is a *synchronization bug*
- The state of the game is different for different players



(a) Snapshot of $\texttt{Player}_i$'s Game.

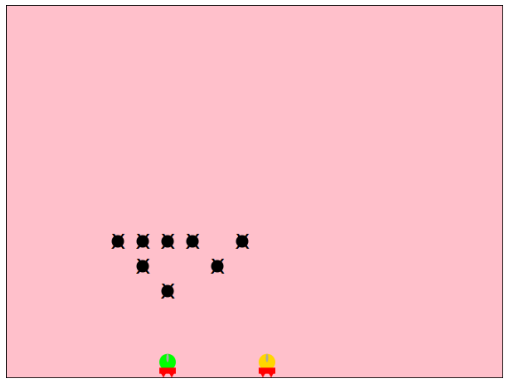


(b) Snapshot of $\texttt{Player}_j$'s Game.

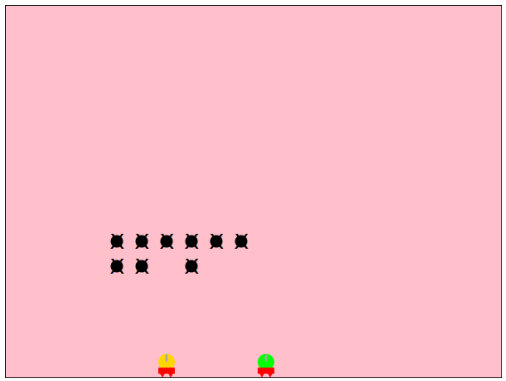
Aliens Attack Version 7

A Subtle Bug

- Make sure you have at least two copies of the player's program saved (each with a unique `MY-NAME` value)
- To play the game as multiple players on your machine remember to first call `run-server` in the server file and then `run` for each of the player files.
- What do you notice, if anything, after playing the game a few times on your computer?
- If you run it enough times you are bound to see that there is a *synchronization bug*
- The state of the game is different for different players



(a) Snapshot of $\texttt{Player}_i$'s Game.



(b) Snapshot of $\texttt{Player}_j$'s Game.

- The army of aliens is different
- How is this possible?

Aliens Attack Version 7

A Subtle Bug

- The problem is due to each player updating its `world` independently
- Messages take time to travel from a player to the server and then to a receiving player

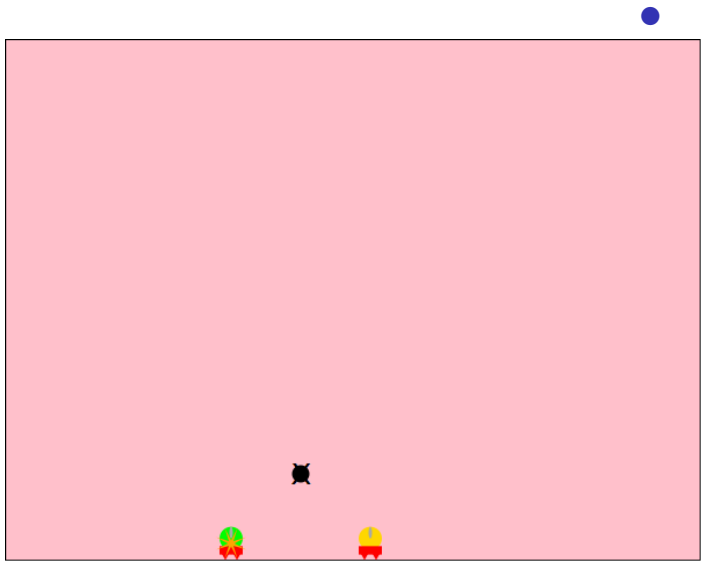
Aliens Attack Version 7

A Subtle Bug

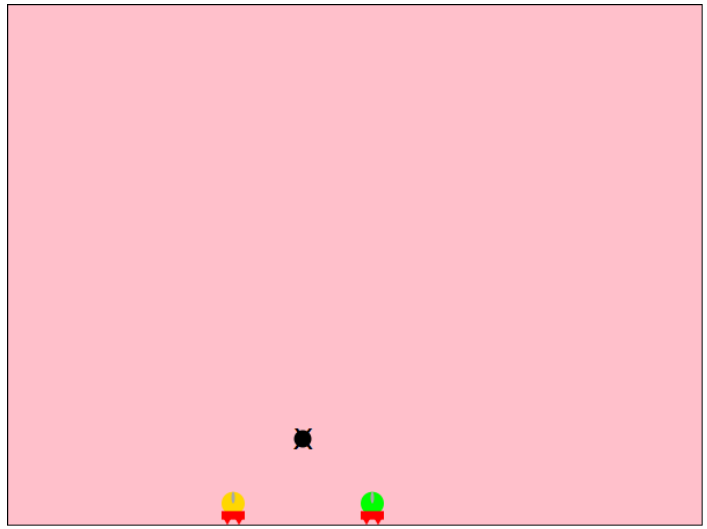
- The problem is due to each player updating its `world` independently
- Messages take time to travel from a player to the server and then to a receiving player
- Consider what happens when $\texttt{Player}_i$ shoots

Aliens Attack Version 7

A Subtle Bug



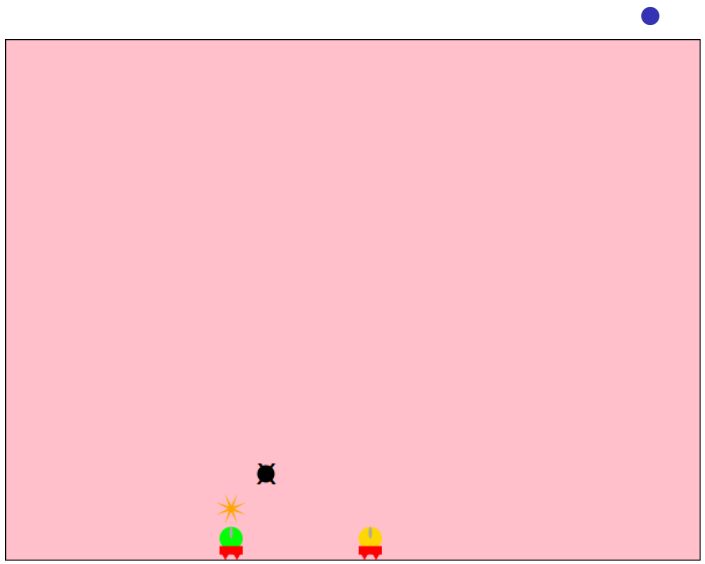
(a) Snapshot of `Player`$_i$'s Game.



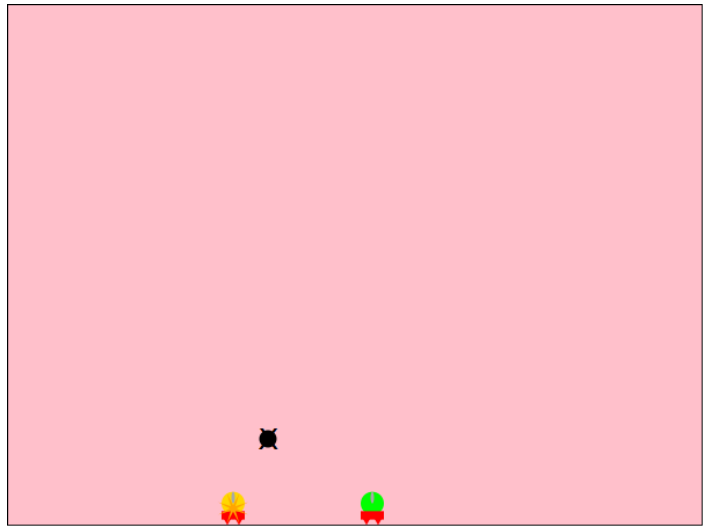
(b) Snapshot of `Player`$_j$'s Game.

Aliens Attack Version 7

A Subtle Bug



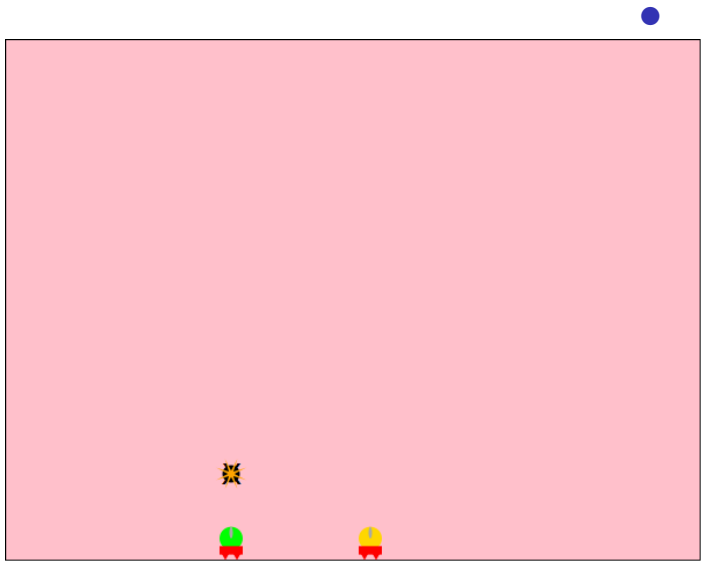
(a) Snapshot of `Player`$_i$'s Game.



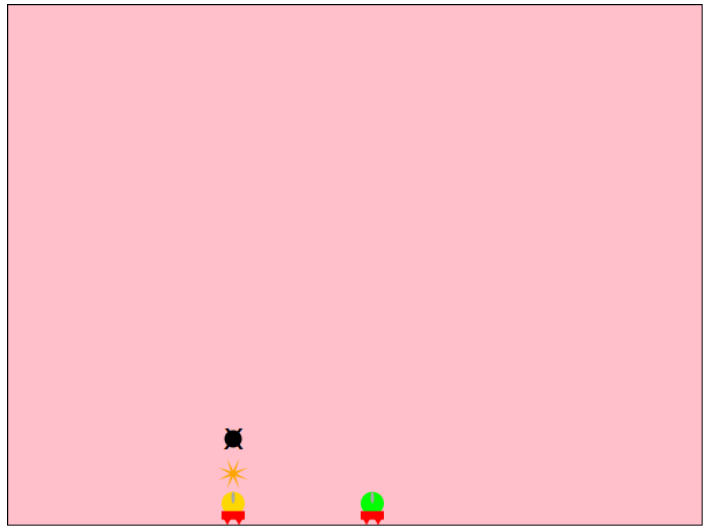
(b) Snapshot of `Player`$_j$'s Game.

Aliens Attack Version 7

A Subtle Bug



(a) Snapshot of `Player`$_i$'s Game.



(b) Snapshot of `Player`$_j$'s Game.

Aliens Attack Version 7

A Subtle Bug

- Synchronization bugs in distributed programming are hard to pinpoint because they do not always manifest themselves
- Sometimes an application may run without seeing the bug and other times the bug is seen
- There is no test we can write to protect ourselves from this potential bug

Aliens Attack Version 7

A Subtle Bug

- Synchronization bugs in distributed programming are hard to pinpoint because they do not always manifest themselves
- Sometimes an application may run without seeing the bug and other times the bug is seen
- There is no test we can write to protect ourselves from this potential bug
- Unfortunately, it is not the only subtle distributed programming bug that may occur
- Another common bug is *deadlock*
- Deadlock occurs when two (or more) components are waiting for each other to perform an action

Aliens Attack Version 7

A Subtle Bug

- What can we do to fix multiplayer Aliens Attack?

Aliens Attack Version 7

A Subtle Bug

- What can we do to fix multiplayer Aliens Attack?
- Given that the problem stems from each player making changes to its own copy of the `world` the solution is to have only one component allowed to make changes to the `world`
- All the other components get the same copy of the `world` whenever it is changed
- The natural choice is to only allow the server to make changes to the `world`
- In this manner maintain all the players synchronized

Aliens Attack Version 7

A Subtle Bug

- What can we do to fix multiplayer Aliens Attack?
- Given that the problem stems from each player making changes to its own copy of the `world` the solution is to have only one component allowed to make changes to the `world`
- All the other components get the same copy of the `world` whenever it is changed
- The natural choice is to only allow the server to make changes to the `world`
- In this manner maintain all the players synchronized
- Does this mean that thin servers are useless?

Aliens Attack Version 7

A Subtle Bug

- What can we do to fix multiplayer Aliens Attack?
- Given that the problem stems from each player making changes to its own copy of the `world` the solution is to have only one component allowed to make changes to the `world`
- All the other components get the same copy of the `world` whenever it is changed
- The natural choice is to only allow the server to make changes to the `world`
- In this manner maintain all the players synchronized
- Does this mean that thin servers are useless?
- No, they are fine when clients do not need to be synchronized or when clients are automatically synchronized by the nature of the application
- For example, thin servers work well for turn-based games

Aliens Attack Version 8

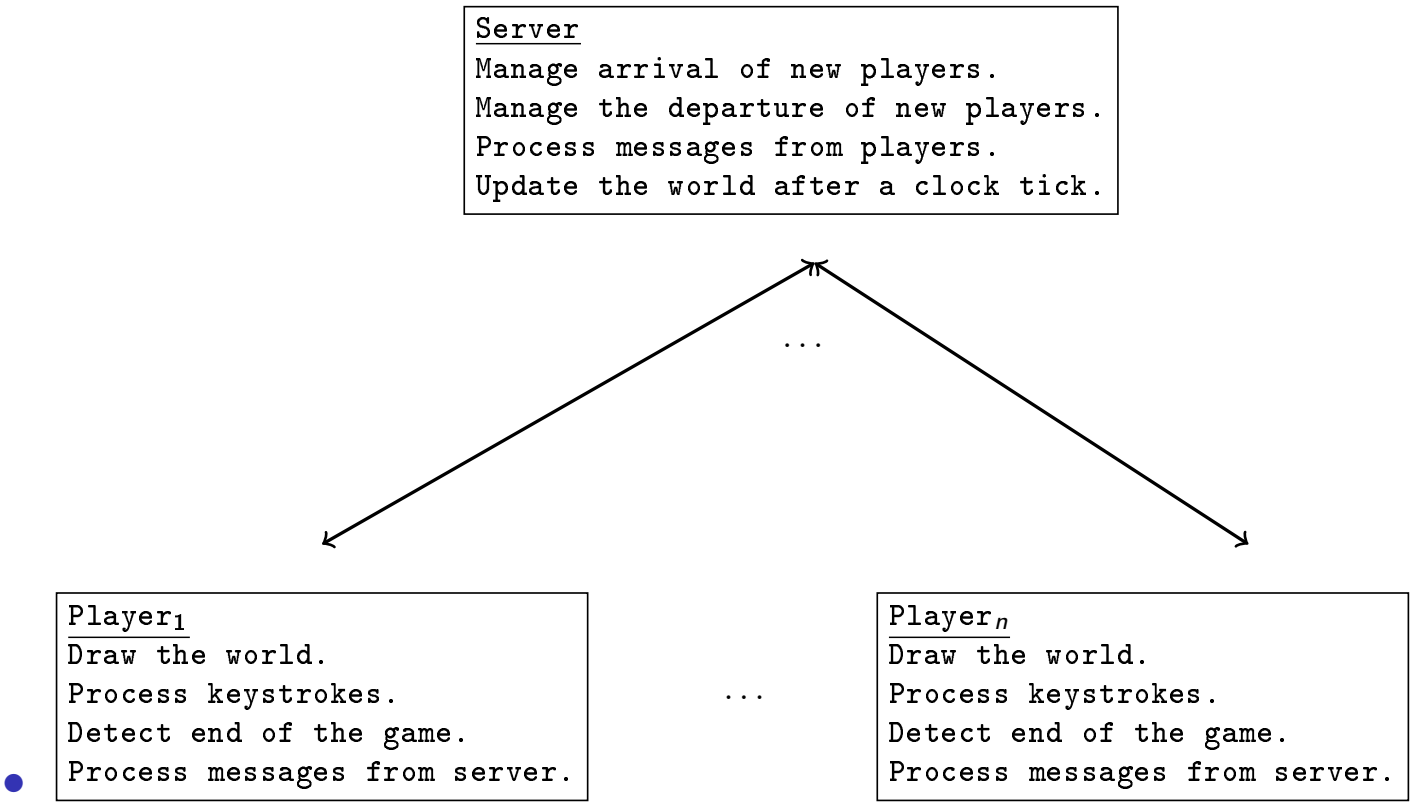
- Multiplayer Aliens Attack using a think server
- Only the server makes changes to the state of the game
- The players do not make changes to their local copy of the `world`.

Aliens Attack Version 8

- Multiplayer Aliens Attack using a think server
- Only the server makes changes to the state of the game
- The players do not make changes to their local copy of the `world`.
- Players still need to have the ability to mover their rocket and to shoot
- When such actions are taken a message is sent to the server, the server creates a new `world`, and the server sends the new `world` to the players
- Given that the server is the only component allowed to update the world, the players cannot become unsynchronized

Aliens Attack Version 8

Components



Aliens Attack Version 8

Data Definitions

- The data definitions for the `world` are the same as those defined for Aliens Attack 6
- Additional `world` for testing:

```
(define WORLD4 (make-world (list (make-ally 8 "iworld3")
                                 (make-ally 5 "iworld2"))
                           (list (make-posn 8 2))
                           'right
                           (list (make-posn 8 4))))
```

Aliens Attack Version 8

Data Definitions

- Server needs to track the players in the game
- Server must maintain the state of the game
- The `world` must be part of the universe

Aliens Attack Version 8

Data Definitions

- Server needs to track the players in the game
- Server must maintain the state of the game
- The `world` must be part of the universe
- ```
  ;; A universe (univ) is a structure:
  ;;     (make-univ (listof iworld) world)
  (define-struct univ (iws game))
  ```
  ```

Aliens Attack Version 8

Data Definitions

- Server needs to track the players in the game
- Server must maintain the state of the game
- The `world` must be part of the universe
- ```
  ;; A universe (univ) is a structure:
  ;;     (make-univ (listof iworld) world)
  (define-struct univ (iws game))
  ```
- ```
  ;; Template for a function on a univ
  #| ;; Sample instances of univ
     (define UNIV1 (make-univ ... ...))

     univ ... → ...
     Purpose:
     (define (f-on-univ a-univ ...)
       (...(f-on-loiw (univ-iws a-univ))...
        ...(f-on-world (univ-world a-univ)...)))

     ;; Sample expressions for f-on-univ
     (define UNIV1-VAL ...) ...

     ;; Tests using sample computations for f-on-univ
     (check-expect (f-on-univ UNIV1 ...) UNIV1-VAL) ...

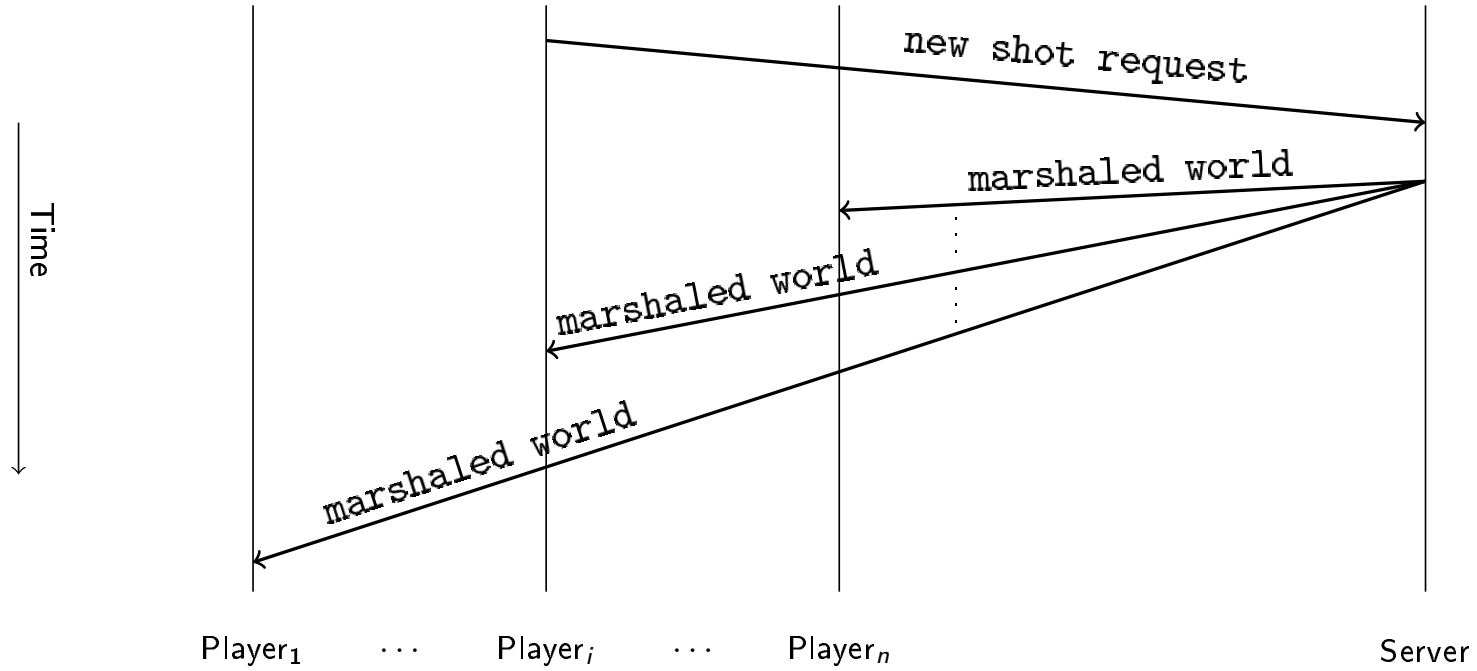
     ;; Tests using sample values for f-on-univ
     (check-expect (f-on-univ ... ...) ...) ...          |#

  ;; Sample instances of universe
  (define INIT-UNIV  (make-univ '() UNINIT-WORLD))
  (define OTHR-UNIV  (make-univ (list iworld1 iworld2) WORLD3))
  (define OTHR-UNIV2 (make-univ (list iworld3 iworld2) WORLD4))
  ```

Aliens Attack Version 8

Communication Protocol: Player Sparked

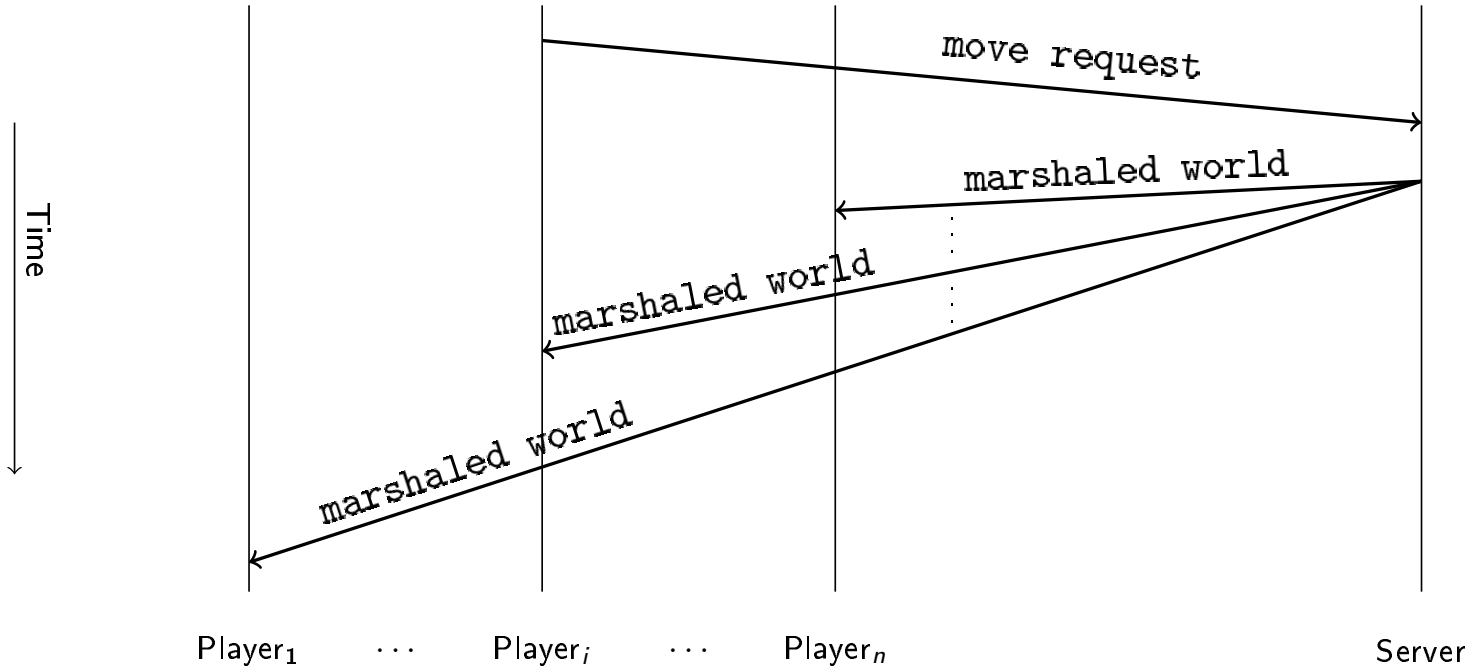
- Player soots



Aliens Attack Version 8

Communication Protocol: Player Sparked

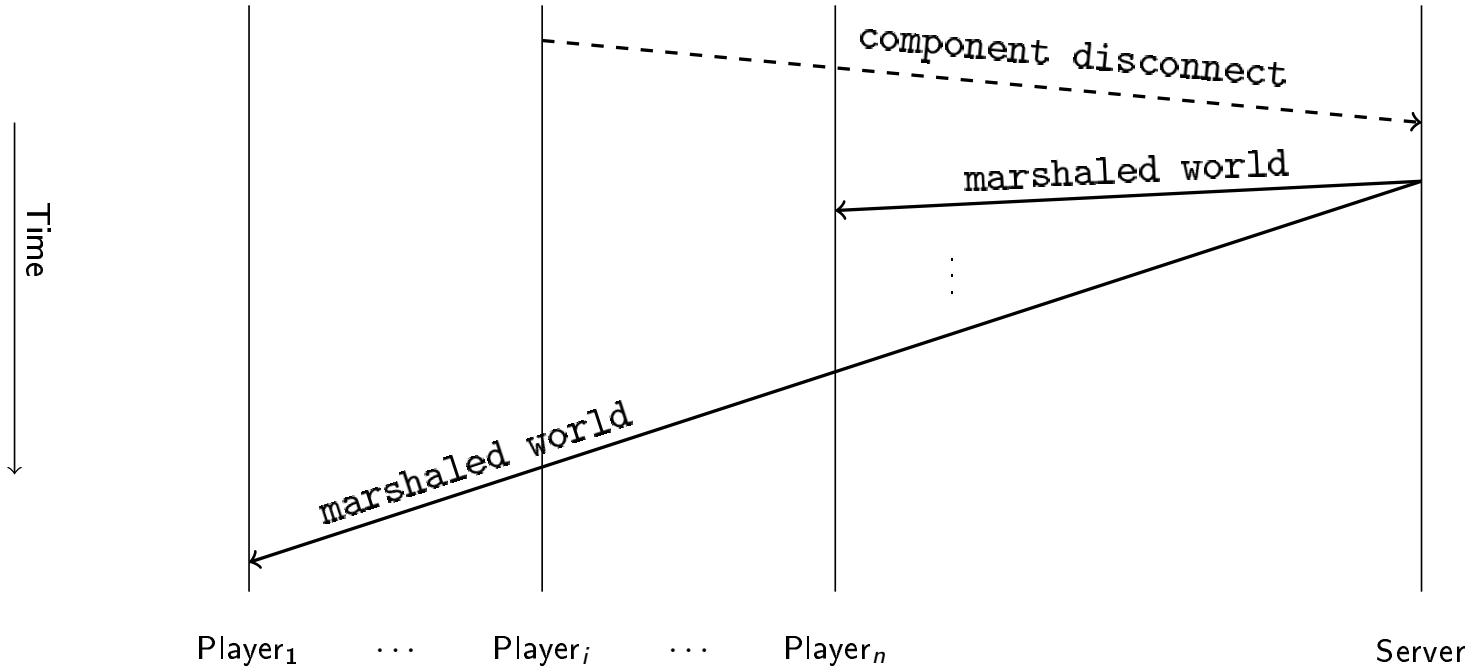
- Player moves rocket



Aliens Attack Version 8

Communication Protocol: Server Sparked

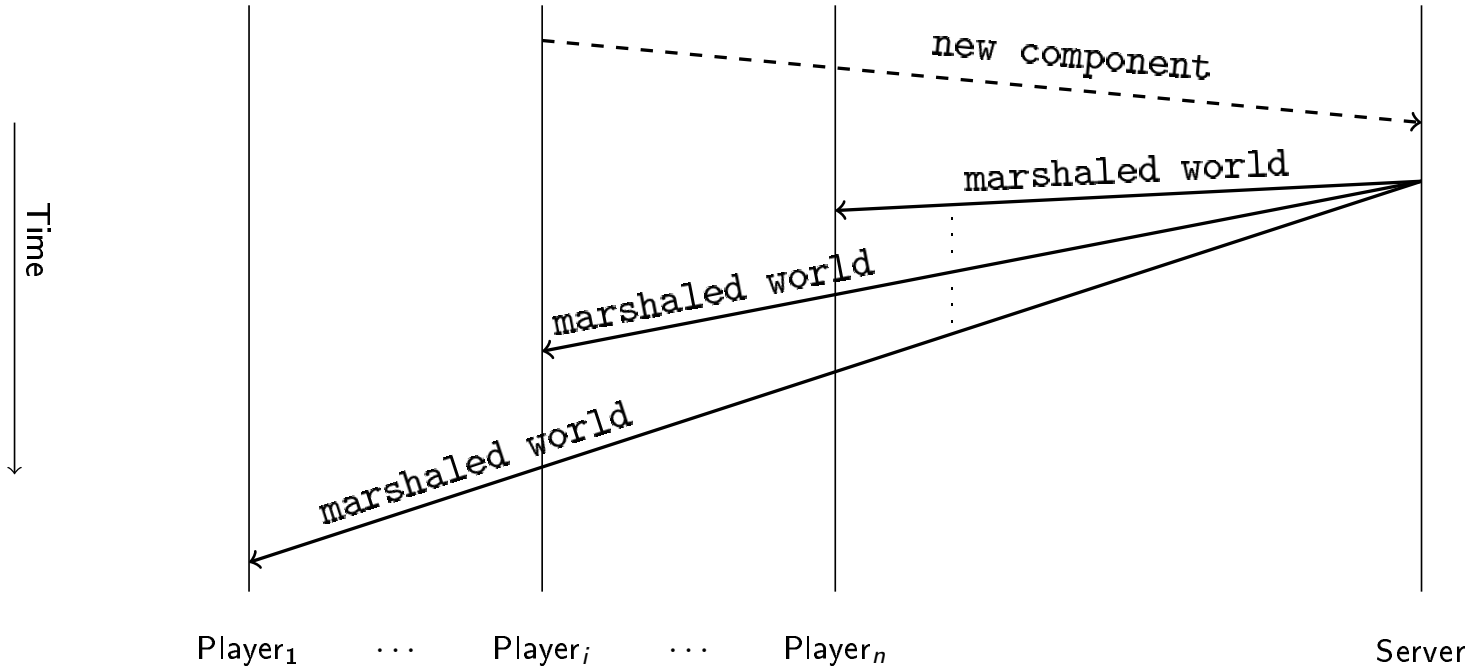
- Player leaves game



Aliens Attack Version 8

Communication Protocol: Server Sparked

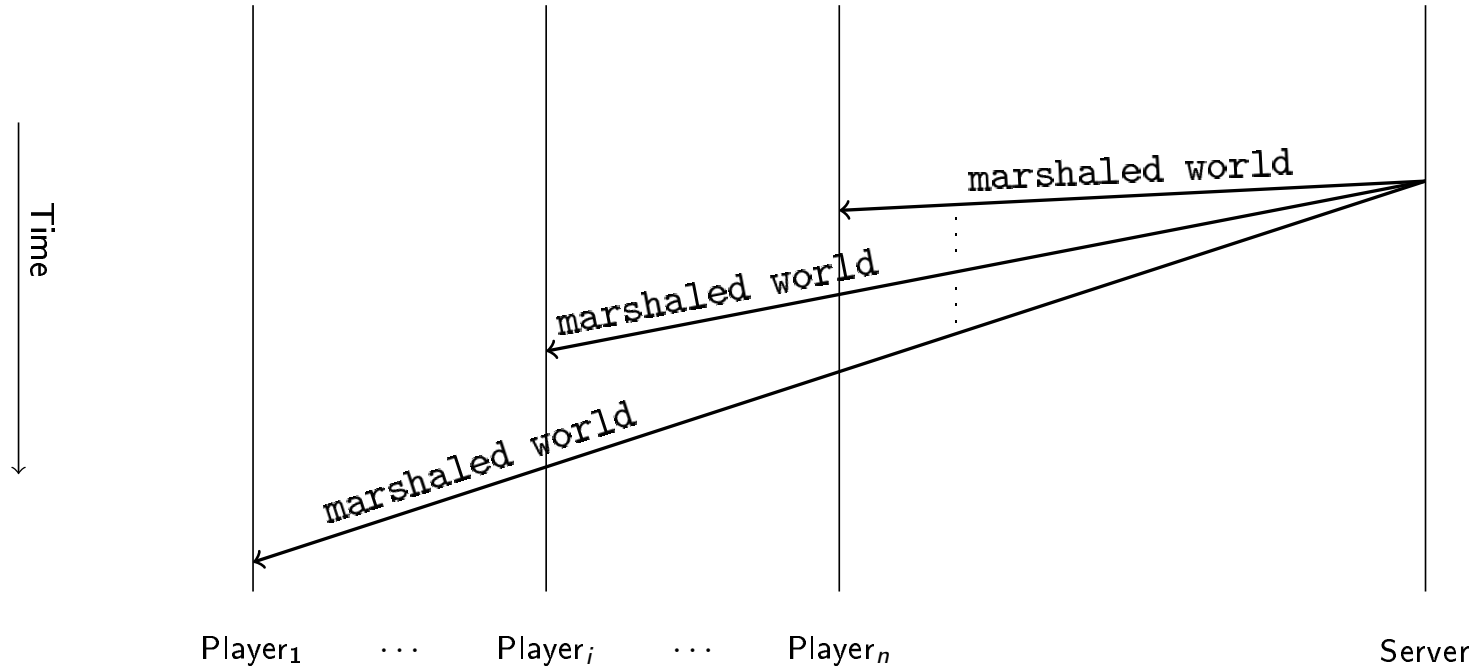
- Player joins the game



Aliens Attack Version 8

Communication Protocol: Server Sparked

- Clock tick



Aliens Attack Version 8

Message Data Definitions

- `#| ;; A to-player message (tpm) is: (cons 'world mw)`

Aliens Attack Version 8

Message Data Definitions

```
#|  ;; A to-player message (tpm) is: (cons 'world mw)

    tpm ... → ...
    Purpose:
    (define (f-on-tpm a-tpm ...)
       (...(f-on-mw (rest a-tpm) ...)...))

    Sample instances of tpm
    (define A-TPM (cons 'world ...))

    Sample expressions for f-on-tpm
    (define A-TPM-VAL (f-on-mw (rest A-TPM) ...))

    Tests using sample for computations for f-on-mw
    (check-expect  (f-on-tpm A-TPM ...) A-TPM-VAL) ...

    Tests using sample for values for f-on-mw
    (check-expect  (f-on-tpm ... ...) ...) ...     |#

    ;; Sample instances of tpm
    (define TPM1 (cons 'world
                       (list (list (list 10 "San Martin"))
                             (list (list 5  8))
                             'right
                             (list (list 12 4)))))

    (define TPM2 (cons 'world
                       (list (list (list 0 "Juarez"))
                             (list (list 5 3))
                             'right
                             (list (list 7 7)))))
```

Aliens Attack Version 8

Message Data Definitions

- Two types of to-server messages: new shot and new move

Aliens Attack Version 8

Message Data Definitions

- Two types of to-server messages: new shot and new move
- ```
  #|  A to-server message (tsm) is either:
      1. (list 'move key)
      2. (list 'shoot)
  ```
  ```

Aliens Attack Version 8

Message Data Definitions

- Two types of to-server messages: new shot and new move
- ```
  #|  A to-server message (tsm) is either:
       1. (list 'move key)
       2. (list 'shoot)
  ```
- ```
  ;; tsm ... → ...   Purpose:
  (define (f-on-tsm a-tsm ...)
    (local [(define tag (first a-tsm))]
      (cond [(eq? tag 'move)  ...]
            [(eq? tag 'shoot) ...]
            [else
              (error
               (format "Unknown to-server message type ~s"
               a-tsm))])))

  ;; Sample instances of tsm
  (define MV-TSM ...)
  (define SH-TSM ...) ...

  ;; Sample expressions for f-on-tsm
  (define MV-TSM-VAL ...)
  (define SH-TSM-VAL ...)

  ;; Tests using sample computations for f-on-tsm
  (check-expect (f-on-tsm MV-TSM ...) MV-TSM-VAL)
  (check-expect (f-on-tsm SH-TSM ...) SH-TSM-VAL) ...
  ;; Tests using sample values for f-on-tsm
  (check-expect (f-on-tsm ... ...) ...) ...     |#

  ;; Sample instances of tsm
  (define MV-LEFT (list 'move "left"))
  (define MV-RGHT (list 'move "right"))
  (define SHOOT   (list 'shoot))
  ```

Aliens Attack Version 8

Marshalling and Unmarshalling

- `world` data definition is unchanged from Aliens Attack version 7
- May use same marshalling and unmarshalling functions

Aliens Attack Version 8

Marshalling and Unmarshalling

- `world` data definition is unchanged from Aliens Attack version 7

- `tsms` only contain data that is suitable for transmission
- No need to implement marshalling and unmarshalling functions for these messages

Aliens Attack Version 8

Component Implementation

- One of the goals is to reuse as much of the code as possible from Aliens Attack 6

Aliens Attack Version 8

Component Implementation

- One of the goals is to reuse as much of the code as possible from Aliens Attack 6
- Based on our problem analysis we can outline how to distribute the handlers among the player and server components

Aliens Attack Version 8

Component Implementation

- One of the goals is to reuse as much of the code as possible from Aliens Attack 6
- Based on our problem analysis we can outline how to distribute the handlers among the player and server components
- Player components:

```
1. draw-world
2. game-over? (and draw-last-world)
3. process-key
4. process-message tpms
```

Aliens Attack Version 8

Component Implementation

- One of the goals is to reuse as much of the code as possible from Aliens Attack 6
- Based on our problem analysis we can outline how to distribute the handlers among the player and server components
- Player components:

```
1. draw-world
2. game-over? (and draw-last-world)
3. process-key
4. process-message tpms
```

- run:

```
;; Z → world
;; Purpose: To run the game
(define (run a-z)
  (big-bang
    UNINIT-WORLD
    [on-draw    draw-world]
    [on-key     process-key]
    [stop-when  game-over? draw-last-world]
    [on-receive process-message]
    [name       MY-NAME]
    [register   LOCALHOST]))
```

- No stanza for tick processing

Aliens Attack Version 8

Component Implementation

- The server component needs the following, possibly refined, handlers from Aliens Attack version 6:

```
1. process-tick
2. process-key
3. process-message tsms
```

Aliens Attack Version 8

Component Implementation

- The server component needs the following, possibly refined, handlers from Aliens Attack version 6:

  1. process-tick
  2. process-key
  3. process-message tsms

-
```
;; Z → univ
;; Purpose: To run the server
(define (run-server a-z)
  (local [(define TICK-RATE 1/4)]
    (universe
     INIT-UNIV
     (on-tick       process-tick TICK-RATE)
     (on-msg        process-message)
     (on-new        add-player)
     (on-disconnect rm-player))))
```

Aliens Attack Version 8

Component Implementation: Player

- `draw-world` and `game-over?` no change from Aliens Attack version 6
- The same holds true for the auxiliary function `draw-last-world`

Aliens Attack Version 8

Component Implementation: Player

- `process-key` handler may not update the game: refinement needed

Aliens Attack Version 8

Component Implementation: Player

- `process-key` handler may not update the game: refinement needed
- If the `world` in uninitialized then nothing needs to be done and the existing world is returned
- Otherwise, the key pressed must be processed

Aliens Attack Version 8

Component Implementation: Player

- `process-key` handler may not update the game: refinement needed
- If the `world` in uninitialized then nothing needs to be done and the existing world is returned
- Otherwise, the key pressed must be processed
- If the key pressed is `"left"` or `"right"` then according to the protocol design a move request must be sent to the server
- Similarly, if the key pressed is `" "` then a new shot request must be sent to the server
- In both cases, a `package` is constructed using the current `world` and the appropriate `tsm`

Aliens Attack Version 8

Component Implementation: Player

- ```
  (check-expect (process-key (make-world
                              (list (make-ally 10 MY-NAME))
                              (list (make-posn 7 2))
                              'right
                              '())
                             "right")
                (make-package (make-world
                               (list (make-ally 10 MY-NAME))
                               (list (make-posn 7 2))
                               'right
                               '())
                              (list 'move "right")))
  (check-expect (process-key (make-world
                              (list (make-ally 10 MY-NAME))
                              (list (make-posn 7 2))
                              'right
                              '())
                             "left")
                (make-package (make-world
                               (list (make-ally 10 MY-NAME))
                               (list (make-posn 7 2))
                               'right
                               '())
                              (list 'move "left")))
  ```
  ```

Aliens Attack Version 8

Component Implementation: Player

- ```
  (check-expect (process-key (make-world
                              (list (make-ally 10 MY-NAME))
                              (list (make-posn 7 2))
                              'left
                              '())
                             " ")
                (make-package (make-world
                               (list (make-ally 10 MY-NAME))
                               (list (make-posn 7 2))
                               'left
                               '())
                              (list 'shoot)))

  (check-expect (process-key (make-world
                              (list (make-ally 10 MY-NAME))
                              (list (make-posn 7 2))
                              'right
                              '())
                             "d")
                (make-world (list (make-ally 10 MY-NAME))
                            (list (make-posn 7 2))
                            'right
                            '()))
  ```
  ```

Aliens Attack Version 8

Component Implementation: Player

```
;; world key → world or package
;; Purpose: Process a key event to return next world
(define (process-key a-world a-key)
  (local
   [;; world key → world
    ;; Purpose: Process a key event to return next world
    ;; ASSUMPTION: The given world is a structure
    (define (process-key a-world a-key)
      (cond [(or (key=? a-key "right")
                 (key=? a-key "left"))
             (make-package a-world (list 'move a-key))]
            [(key=? a-key " ")
             (make-package a-world (list 'shoot))]
            [else a-world]))]
   (if (eq? a-world UNINIT-WORLD)
       a-world
       (process-key a-world a-key))))
```

Aliens Attack Version 8

Component Implementation: Player

- `process-message` needs to process the single variety of `tpm` that has a marshaled `world`

Aliens Attack Version 8

Component Implementation: Player

- `process-message` needs to process the single variety of `tpm` that has a marshaled `world`

- ```
  ;; Sample expressions for process-message
  (define PM-TPM1 (unmarshal-world (rest TPM1)))
  (define PM-TPM2 (unmarshal-world (rest TPM2)))
  ```
  ```

Aliens Attack Version 8

Component Implementation: Player

- `process-message` needs to process the single variety of `tpm` that has a marshaled `world`
- `;; world tpm → world  Purpose: Update world with the given tpm`
  `(define (process-message a-world a-tpm)`

- `;; Sample expressions for process-message`
  `(define PM-TPM1 (unmarshal-world (rest TPM1)))`
  `(define PM-TPM2 (unmarshal-world (rest TPM2)))`

Aliens Attack Version 8

Component Implementation: Player

- process-message needs to process the single variety of tpm that has a marshaled world
- ;; world tpm → world Purpose: Update world with the given tpm
```
(define (process-message a-world a-tpm)
```
- ;; Sample expressions for process-message
```
(define PM-TPM1 (unmarshal-world (rest TPM1)))
(define PM-TPM2 (unmarshal-world (rest TPM2)))
```
- ;; Tests using sample computations for process-message
```
(check-expect (process-message INIT-WORLD  TPM1) PM-TPM1)
(check-expect (process-message INIT-WORLD2 TPM2) PM-TPM2)
;; Tests using sample values for process-message
(check-expect
 (process-message WORLD3 (list 'world
                               (list (list 9 "Bolivar"))
                               (list (list 7 2))
                               'left
                               '()))
 (make-world (list (make-ally 9 "Bolivar"))
                   (list (make-posn 7 2))
                   'left
                   '()))
```

Aliens Attack Version 8

Component Implementation: Player

- process-message needs to process the single variety of tpm that has a marshaled world
- ;; world tpm → world Purpose: Update world with the given tpm
  (define (process-message a-world a-tpm)
- (unmarshal-world (rest a-tpm)))
- ;; Sample expressions for process-message
  (define PM-TPM1 (unmarshal-world (rest TPM1)))
  (define PM-TPM2 (unmarshal-world (rest TPM2)))
- ;; Tests using sample computations for process-message
  (check-expect (process-message INIT-WORLD TPM1) PM-TPM1)
  (check-expect (process-message INIT-WORLD2 TPM2) PM-TPM2)
  ;; Tests using sample values for process-message
  (check-expect
   (process-message WORLD3 (list 'world
                                 (list (list 9 "Bolivar"))
                                 (list (list 7 2))
                                 'left
                                 '()))
   (make-world (list (make-ally 9 "Bolivar"))
                     (list (make-posn 7 2))
                     'left
                     '()))

Aliens Attack Version 8

Component Implementation: Server

- The tick handler must take as input a `universe` and return a `universe`

Aliens Attack Version 8

Component Implementation: Server

- The tick handler must take as input a `universe` and return a `universe`
- Signature is different from the `process-tick` function from Aliens Attack version 6

Aliens Attack Version 8

Component Implementation: Server

- The tick handler must take as input a `universe` and return a `universe`
- Signature is different from the `process-tick` function from Aliens Attack version 6
- `INIT-UNIV`: no player has joined and universe remains unchanged

Aliens Attack Version 8

Component Implementation: Server

- The tick handler must take as input a `universe` and return a `universe`
- Signature is different from the `process-tick` function from Aliens Attack version 6
- `INIT-UNIV`: no player has joined and universe remains unchanged
- Otherwise, the universe ought to be updated
- The list of `iworlds` remains unchanged
- The state of the `game` needs to be updated by calling `process-tick` from Aliens Attack version 6
- By communication protocol, the new `world` value must be sent to all the players.

Aliens Attack Version 8

Component Implementation: Server

- The tick handler must take as input a `universe` and return a `universe`
- Signature is different from the `process-tick` function from Aliens Attack version 6
- `INIT-UNIV`: no player has joined and universe remains unchanged
- Otherwise, the universe ought to be updated
- The list of `iworlds` remains unchanged
- The state of the `game` needs to be updated by calling `process-tick` from Aliens Attack version 6
- By communication protocol, the new `world` value must be sent to all the players.
- The handler is outlined as follows:

```
;; univ → bundle  Purpose: Create a new universe after a clock tick
(define (process-tick a-univ)
  (local [...
          ;; world → world  Purpose: Create new world after a clock tick
          ;; ASSUMPTION: The world is a structure
          (define (process-tick a-world) ... )]
    (if (equal? a-univ INIT-UNIV)
        (make-bundle a-univ '() '())
        (local [(define new-game (process-tick (univ-game a-univ)))]
          (make-bundle
           (make-univ (univ-iws a-univ) new-game)
           (map (λ (iw)
                  (make-mail
                    iw
                    (cons 'world (marshal-world new-game))))
                (univ-iws a-univ))
           '())))))
```

Aliens Attack Version 8

Component Implementation: Server

- The tests illustrate that changes to the `world` are correctly made and that the list of `iworlds` remains unchanged

```
;; Tests using sample values for process-tick
(check-expect
  (process-tick
    (make-univ (list iworld1 iworld3)
               (make-world (list (make-ally 9 "iworld1") (make-ally 2 "iworld3"))
                           (list (make-posn 2 5))
                           'left
                           (list (make-posn 3 6) NO-SHOT))))
    (make-bundle
     (make-univ (list iworld1 iworld3)
                (make-world (list (make-ally 9 "iworld1") (make-ally 2 "iworld3"))
                            (list (make-posn 1 5))
                            'left
                            (list (make-posn 3 5))))
     (list (make-mail iworld1
                      (list 'world
                            (list (list 9 "iworld1") (list 2 "iworld3"))
                            (list (list 1 5))
                            'left
                            (list (list 3 5))))
           (make-mail iworld3
                      (list 'world
                            (list (list 9 "iworld1") (list 2 "iworld3"))
                            (list (list 1 5))
                            'left
                            (list (list 3 5)))))
     '()))
```

More tests in the textbook

Aliens Attack Version 8

Component Implementation: Server

- `process-message` is written by specializing the template for functions on a `tsm`

Aliens Attack Version 8

Component Implementation: Server

- `process-message` is written by specializing the template for functions on a `tsm`
- If the given `tsm`'s tag is `shoot` or `move` then the `universe`'s game needs to be updated
- Refinie `process-key` from Aliens Attack version 6
- The code for `process-key` always moves the `ally` or creates a `shot` for a single player
- In this version of the game the server needs to do so for an arbitrary player

Aliens Attack Version 8

Component Implementation: Server

- `process-message` is written by specializing the template for functions on a `tsm`
- If the given `tsm`'s tag is `shoot` or `move` then the `universe`'s game needs to be updated
- Refinie `process-key` from Aliens Attack version 6
- The code for `process-key` always moves the `ally` or creates a `shot` for a single player
- In this version of the game the server needs to do so for an arbitrary player
- If `process-key` is made local to `process-message` then the name of the `iworld` making the request and the `universe`'s `world` value are in scope and may be used to correctly move or shoot
- This means that `process-key` only needs the `key` to process as input
- The value returned by `process-key`, therefore, must be a bundle that has a `univ` with the new `world` value and the `mails` containing this new value for the players

Aliens Attack Version 8

Component Implementation: Server

- ```
  ;; univ iworld tsm → bundle throws error
  ;; Purpose: Process the message to create new universe
  ;; ASSUMPTION: The given univ is not INIT-UNIV
  (define (process-message a-univ an-iw a-tsm)
  ```
  ```

Aliens Attack Version 8

Component Implementation: Server

```
;; univ iworld tsm → bundle throws error
;; Purpose: Process the message to create new universe
;; ASSUMPTION: The given univ is not INIT-UNIV
(define (process-message a-univ an-iw a-tsm)
 (local [(define tag (first a-tsm)) (define name (iworld-name an-iw))
         (define game (univ-game a-univ))
         ;; key → bundle  Purpose: Create a bundle for player request
         (define (process-key a-key)
```

Aliens Attack Version 8

Component Implementation: Server

- ```
  ;; univ iworld tsm → bundle throws error
  ;; Purpose: Process the message to create new universe
  ;; ASSUMPTION: The given univ is not INIT-UNIV
  (define (process-message a-univ an-iw a-tsm)
  ```
- ```
   (local [(define tag (first a-tsm)) (define name (iworld-name an-iw))
           (define game (univ-game a-univ))
           ;; key → bundle  Purpose: Create a bundle for player request
           (define (process-key a-key)
  ```

- ```
      (if (or (eq? tag 'shoot) (eq? tag 'move))
          (process-key (if (eq? tag 'shoot) " " (second a-tsm)))
          (error (format "Unknown to-server message type ~s" a-tsm)))))
  ```

Tests in textbook
  ```

Aliens Attack Version 8

Component Implementation: Server

```
;; univ iworld tsm → bundle throws error
;; Purpose: Process the message to create new universe
;; ASSUMPTION: The given univ is not INIT-UNIV
(define (process-message a-univ an-iw a-tsm)
 (local [(define tag (first a-tsm)) (define name (iworld-name an-iw))
         (define game (univ-game a-univ))
         ;; key → bundle  Purpose: Create a bundle for player request
         (define (process-key a-key)
         (local [   ... All local functions from before
                 (define nw
                  (make-world
                   (cond [(key=? a-key "right")
                          (move-ally-right name (world-allies game))]
                         [(key=? a-key "left")
                          (move-ally-left name (world-allies game))]
                         [else (world-allies game)])
                   (world-aliens game)
                   (world-dir game)
                   (if (key=? a-key " ")
                       (cons (process-shooting
                              (ally-rocket (get-ally name (world-allies game))))
                             (world-shots game))
                       (world-shots game))))]
```

```
   (if (or (eq? tag 'shoot) (eq? tag 'move))
       (process-key (if (eq? tag 'shoot) " " (second a-tsm)))
       (error (format "Unknown to-server message type ~s" a-tsm)))))
```

Tests in textbook

Aliens Attack Version 8

Component Implementation: Server

```
;; univ iworld tsm → bundle throws error
;; Purpose: Process the message to create new universe
;; ASSUMPTION: The given univ is not INIT-UNIV
(define (process-message a-univ an-iw a-tsm)
 (local [(define tag (first a-tsm)) (define name (iworld-name an-iw))
         (define game (univ-game a-univ))
         ;; key → bundle  Purpose: Create a bundle for player request
         (define (process-key a-key)
         (local [   ... All local functions from before
                 (define nw
                  (make-world
                   (cond [(key=? a-key "right")
                          (move-ally-right name (world-allies game))]
                         [(key=? a-key "left")
                          (move-ally-left name (world-allies game))]
                         [else (world-allies game)])
                   (world-aliens game)
                   (world-dir game)
                   (if (key=? a-key " ")
                       (cons (process-shooting
                              (ally-rocket (get-ally name (world-allies game))))
                             (world-shots game))
                       (world-shots game))))]
           (make-bundle
            (make-univ (univ-iws a-univ) nw)
            (map (λ (iw) (make-mail iw (cons 'world (marshal-world nw))))
                 (univ-iws a-univ))
            '()))])]  Tests in textbook
   (if (or (eq? tag 'shoot) (eq? tag 'move))
       (process-key (if (eq? tag 'shoot) " " (second a-tsm)))
       (error (format "Unknown to-server message type ~s" a-tsm)))))  Tests in textbook
```

Aliens Attack Version 8

Component Implementation: Server

- The handler to add new players must reject `iworlds` that have a name that is already in use
- If it is the first player to join the initial `world` containing the `ally` that just joined is used to create a new `universe`
- Otherwise, the new `ally` is added to the existing game

Aliens Attack Version 8

Component Implementation: Server

- ```
  ;; Sample expressions for add-player
  (define RPT-ADD (make-bundle OTHR-UNIV '() (list iworld1)))
  ```
  ```

Aliens Attack Version 8

Component Implementation: Server

- ```
  ;; Sample expressions for add-player
  (define RPT-ADD (make-bundle OTHR-UNIV '() (list iworld1)))
  ```
- ```
  (define EMP-ADD
   (local [(define new-iws (cons iworld2 (univ-iws INIT-UNIV)))
           (define game (univ-game INIT-UNIV))
           (define new-game
             (if (equal? game UNINIT-WORLD)
                 (make-world (list (make-ally INIT-ROCKET (iworld-name iworld2)))
                             INIT-LOA INIT-DIR INIT-LOS)
                 (make-world (cons (make-ally INIT-ROCKET (iworld-name iworld2))
                                   (world-allies game))
                             (world-aliens game) (world-dir game) (world-shots game))))]
    (make-bundle (make-univ new-iws new-game)
                 (map (λ (iw) (make-mail iw (cons 'world (marshal-world new-game))))
                      new-iws)
                 '())))
  ```

Aliens Attack Version 8

Component Implementation: Server

- ```
  ;; Sample expressions for add-player
  (define RPT-ADD (make-bundle OTHR-UNIV '() (list iworld1)))
  ```
- ```
  (define EMP-ADD
   (local [(define new-iws (cons iworld2 (univ-iws INIT-UNIV)))
           (define game (univ-game INIT-UNIV))
           (define new-game
             (if (equal? game UNINIT-WORLD)
                 (make-world (list (make-ally INIT-ROCKET (iworld-name iworld2)))
                             INIT-LOA INIT-DIR INIT-LOS)
                 (make-world (cons (make-ally INIT-ROCKET (iworld-name iworld2))
                                   (world-allies game))
                             (world-aliens game) (world-dir game) (world-shots game))))]
    (make-bundle (make-univ new-iws new-game)
                 (map (λ (iw) (make-mail iw (cons 'world (marshal-world new-game))))
                      new-iws)
                 '())))
  ```
- ```
  (define NEW-ADD
   (local [(define new-iws (cons iworld3 (univ-iws OTHR-UNIV)))
           (define game (univ-game OTHR-UNIV))
           (define new-game
             (if (equal? game UNINIT-WORLD)
                 (make-world (list (make-ally INIT-ROCKET (iworld-name iworld3)))
                             INIT-LOA INIT-DIR INIT-LOS)
                 (make-world (cons (make-ally INIT-ROCKET (iworld-name iworld3))
                                   (world-allies game))
                             (world-aliens game) (world-dir game) (world-shots game))))]
    (make-bundle (make-univ new-iws new-game)
                 (map (λ (iw) (make-mail iw (cons 'world (marshal-world new-game))))
                      new-iws)
                 '())))
  ```
  ```

Aliens Attack Version 8

Component Implementation: Server

```
;; universe iworld → bundle
;; Purpose: Add new world to the universe
(define (add-player a-univ an-iw)
  (if (member? (iworld-name an-iw) (map iworld-name (univ-iws a-univ)))
      (make-bundle a-univ '() (list an-iw))
      (local [(define new-iws (cons an-iw (univ-iws a-univ)))
              (define game (univ-game a-univ))
              (define new-game (if (equal? game UNINIT-WORLD)
                                   (make-world
                                    (list (make-ally INIT-ROCKET (iworld-name an-iw)))
                                    INIT-LOA
                                    INIT-DIR
                                    INIT-LOS)
                                   (make-world
                                    (cons (make-ally INIT-ROCKET (iworld-name an-iw))
                                          (world-allies game))
                                    (world-aliens game)
                                    (world-dir game)
                                    (world-shots game))))]
        (make-bundle
         (make-univ new-iws new-game)
         (map (λ (iw) (make-mail iw (cons 'world (marshal-world new-game))))
              new-iws)
         '()))))
```

Aliens Attack Version 8

Component Implementation: Server

- Remove a player: create a new `world` and new (listof iws)
- Communication protocol: send `world` to all the (remaining) players

Aliens Attack Version 8

Component Implementation: Server

- Remove a player: create a new `world` and new (listof iws)
- Communication protocol: send `world` to all the (remaining) players
-
```
;; Sample expressions for rm-player
(define RM-IW1
 (local
   [(define iws  (univ-iws OTHR-UNIV))  (define game (univ-game OTHR-UNIV))
    (define new-iws
      (filter (λ (iw) (not (string=? (iworld-name iworld1) (iworld-name iw)))) iws))
    (define new-game
      (make-world (filter (λ (a) (not (string=? (iworld-name iworld1) (ally-name a))))
                          (world-allies game))
                  (world-aliens game) (world-dir game) (world-shots game)))]
   (make-bundle (make-univ new-iws new-game)
                (map (λ (iw)
                       (make-mail iw (cons 'world (marshal-world new-game))))
                     new-iws)
                '())))
(define RM-IW2
 (local [(define iws  (univ-iws OTHR-UNIV2))  (define game (univ-game OTHR-UNIV2))
         (define new-iws
          (filter (λ (iw) (not (string=? (iworld-name iworld2) (iworld-name iw)))) iws))
         (define new-game
           (make-world (filter (λ (a)
                                 (not (string=? (iworld-name iworld2) (ally-name a))))
                               (world-allies game))
                       (world-aliens game) (world-dir game) (world-shots game)))]
    (make-bundle
     (make-univ new-iws new-game)
     (map (λ (iw) (make-mail iw (cons 'world (marshal-world new-game)))) new-iws)
     '())))
```

Aliens Attack Version 8

Component Implementation: Server

```
;; univ iworld → bundle  Purpose: Remove given iw from universe and game
;; ASSUMPTION: Given univ is not INIT-UNIV
(define (rm-player a-univ an-iw)
 (local [(define iws  (univ-iws a-univ))  (define game (univ-game a-univ))
         (define new-iws (filter
                          (λ (iw) (not (string=? (iworld-name an-iw) (iworld-name iw))))
                          iws))
         (define new-game (make-world
                           (filter
                            (λ (a) (not (string=? (iworld-name an-iw) (ally-name a))))
                            (world-allies game))
                           (world-aliens game)
                           (world-dir game)
                           (world-shots game)))]
    (make-bundle (make-univ new-iws new-game)
                 (map (λ (iw) (make-mail iw (cons 'world (marshal-world new-game))))
                      new-iws)
                 '())))
;; Tests using sample computations for rm-player
(check-expect (rm-player OTHR-UNIV  iworld1) RM-IW1)
(check-expect (rm-player OTHR-UNIV2 iworld2) RM-IW2)
;; Tests using sample computations for rm-player
(check-expect (rm-player
               (make-univ
                (list iworld3)
                (make-world (list (make-ally 8 "iworld3"))  '() 'down '()))
               iworld3)
              (make-bundle (make-univ '() (make-world '() '() 'down '())) '() '()))
```

Aliens Attack Version 8

A Subtle Problem

- Play the game with one or two friends
- Assuming your internet connection is fast enough the game ought to run smoothly

Aliens Attack Version 8

A Subtle Problem

- Play the game with one or two friends
- Assuming your internet connection is fast enough the game ought to run smoothly
- As the number of players increases the game may become choppy
- This may be due to *communication overhead*
- Communication overhead is the proportion of time that is spent exchanging messages instead of advancing the state of the game
- If communication overhead is large enough the game becomes slow.

Aliens Attack Version 8

A Subtle Problem

- Play the game with one or two friends
- Assuming your internet connection is fast enough the game ought to run smoothly
- As the number of players increases the game may become choppy
- This may be due to *communication overhead*
- Communication overhead is the proportion of time that is spent exchanging messages instead of advancing the state of the game
- If communication overhead is large enough the game becomes slow.
- In distributed programming communication is necessary and, therefore, some degree of communication overhead is necessary
- It is desirable, however, to keep this overhead small
- There is no universal solution to this problem
- Common approaches to reduce communication overhead are to limit the number messages exchanged and reduce the size of the messages exchanged

Aliens Attack Version 8

A Subtle Problem

- Play the game with one or two friends
- Assuming your internet connection is fast enough the game ought to run smoothly
- As the number of players increases the game may become choppy
- This may be due to *communication overhead*
- Communication overhead is the proportion of time that is spent exchanging messages instead of advancing the state of the game
- If communication overhead is large enough the game becomes slow.
- In distributed programming communication is necessary and, therefore, some degree of communication overhead is necessary
- It is desirable, however, to keep this overhead small
- There is no universal solution to this problem
- Common approaches to reduce communication overhead are to limit the number messages exchanged and reduce the size of the messages exchanged
- In Aliens Attack 8, the `world` value is always sent to the players
- This is done despite the fact that only one component of the `world` has changed
- A different communication protocol may transmit to the players only the components that have changed
- This may or may not be effective, but is worth trying if communication overhead makes the game slow

Aliens Attack Version 8

FINAL EXAM

- Work in groups of 3-4 people
- You have a choice
- Problem 339: Redesign with a communication protocol that only communicates elements that have changed
- New Game: Alien blockade
  1. Multiple aliens are blockading earth moving horizontally across the scene at different levels
  2. Each players controls a rocket that is attempting to escape and that starts at the bottom
  3. A rocket always moves up and a player can move it left-right
  4. A rocket reaches the top means it escaped
  5. A rocket that hits alien or another ship must start from the bottom again
  6. Game starts when the first player joins
  7. Players can join while the game is in progress
  8. Game ends when all rockets escape

Aliens Attack Version 8

Final words...

- Congratulations!

Aliens Attack Version 8

Final words...

- Congratulations!
- Still much more you can learn about problem solving and programming
- This journey is inevitable even for those that do not aspire to become Computer Scientists
- Remember that problem solving is at the heart of many human activities
- All good things must continue...

Aliens Attack Version 8

Final words...

- Congratulations!
- Still much more you can learn about problem solving and programming
- This journey is inevitable even for those that do not aspire to become Computer Scientists
- Remember that problem solving is at the heart of many human activities
- All good things must continue...
- Computer Science Students:
  - Be patient and apply the skills you have learned in the future
  - Few textbooks on programming emphasize design
  - Learn a new programming language every semester and summer
  - Learn about programming language implementation: central to CS

Aliens Attack Version 8

Final words...

- Congratulations!
- Still much more you can learn about problem solving and programming
- This journey is inevitable even for those that do not aspire to become Computer Scientists
- Remember that problem solving is at the heart of many human activities
- All good things must continue...
- Computer Science Students:
  - Be patient and apply the skills you have learned in the future
  - Few textbooks on programming emphasize design
  - Learn a new programming language every semester and summer
  - Learn about programming language implementation: central to CS
- Non Computer Science Students:
  - You nay feel excited, overwhelmed, or both
  - You are likely to program throughout your life
  - Remember: if you are problem solving then you are programming
  - You may write essays, diagnose a patient, or create a piece of music
  - What do these activities have in common with programming?
  - They process data and are refined until you are satisfied with the result. Is this truly different than finally designing Aliens Attack 8?
  - Use lessons in domains other than programming
  - Realize that problem solving and programming are fully intertwined with life

Aliens Attack Version 8

Final words...

- HAVE A WONDEFUL BREAK!!!