LULEÅ
TEKNISKA
UNIVERSITET

Interface specification of

Game protocol

Lab 4 Network programming

2010-08-30

Short specification

Game start
A Client send Join message to server with id=0 and sequence number=0;
Server replies with a Message head where type= Join and id=clients assigned value.
Server update clients with NewPlayer Change message(s) including info about the new client and it's id.

Event messages
Clients send move request event to Server.
Server may send NewPlayerPosition Change message(s) to connected clients.

Text messages
In game chat messages, id=0 equals "muticast chat".

Remarks

All messages (except Join) shall contain client id and sequence number.

Sequence number must increase for every message sent.

The Server decides actions in response to incoming messages.

Clients are only allowed to move 2D objects in response to incoming NewPlayerPosition messages.
The 2D objects shall represent a figure according to the ObjectForm for each object id.

OPTIONAL: Clients may use other 2D objects than the ObjectForm specified in the protocol.
In that case, use a proper object according to the protocols ObjectDesc.

The game world coordinate system is (-100, -100 to 100,100).

// Enums och constants

```
#define MAXNAMELEN    32


enum ObjectDesc
{
  Human,
  NonHuman,
  Vehicle,
  StaticObject
};


enum ObjectForm
{
  Cube,
  Sphere,
  Pyramid,
  Cone
};

struct Coordinate
{
  int                 x;
  int                 y;
```

// Message head

```
enum MsgType
{
  Join,                          // Client joining game at server

  Leave,                         // Client leaving game

  Change,                        // Information to clients

  Event,                         // Information from clients to server

  TextMessage                    // Send text messages to one or all
};


// Included first in all messages

struct MsgHead
{
  unsigned int        length;    // Total length for whole message
  unsigned int        seq_no;    // Sequence number
  unsigned int        id;        // Client ID or 0;
  MsgType             type;      // Type of message
};
```

// Message type Join (Client -> Server)

```
struct JoinMsg
{
  MsgHead             head;
  ObjectDesc          desc;
  ObjectForm          form;
  char                name[MAXNAMELEN];     // nullterminated!, or empty
};
```

// Message type Leave (Client -> Server)

```
struct LeaveMsg
{
  MsgHead             head;
};
```

// Message type Change (Server -> Client)

```
enum ChangeType
{
  NewPlayer,
  PlayerLeave,
  NewPlayerPosition
};

// Included first in all Change messages

struct ChangeMsg
{
  MsgHead             head;
  ChangeType          type;
};
```

// Variations of ChangeMsg

```
struct NewPlayerMsg
{
  ChangeMsg           msg;       //Change message header with new client id
  ObjectDesc          desc;
  ObjectForm          form;
  char                name[MAXNAMELEN];     // nullterminated!, or empty
};

struct PlayerLeaveMsg
{
  ChangeMsg           msg;       //Change message header with new client id
};

struct NewPlayerPositionMsg
{
  ChangeMsg           msg;       //Change message header
  Coordinate          pos;       //New object position
  Coordinate          dir;       //New object direction
};
```

// Messages of type Event (Client -> Server)

```
enum EventType
{
  Move
};

// Included first in all Event messages

struct EventMsg
{
  MsgHead             head;
  EventType           type;
};
```

// Variantions of EventMsg

```
struct MoveEvent
{
  EventMsg            event;
  Coordinate          pos;       //New object position
  Coordinate          dir;       //New object direction

};
```

// Messages of type TextMessage

```
struct TextMessageMsg            // Optional at client side!!!
{
  MsgHead             head;
  char                text[1];   // NULL-terminerad array av chars.
};
```