$=1$

Visualizer

1.0.0

Generated by Doxygen 1.10.0

Chapter 1

Crossy Road clone

CS202 group project.

1.1 Dependencies

1.1.1 Required

- GCC 11
- Cmake
- Makefile

1.1.2 Automated downloads

- Raylib
- GLFW
- cppyaml

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 PASSETS Namespace Reference

Variables

- const std::string GRAPHIC_ = "assets/graphics/"
- const std::string SOUND_ = "assets/sounds/"
- const std::string FONT_ = "assets/fonts/"

6.1.1 Variable Documentation

6.1.1.1 FONT_

```
const std::string PASSETS::FONT_ = "assets/fonts/"  [extern]
```

Definition at line 6 of file constant.cpp.

6.1.1.2 GRAPHIC_

```
const std::string PASSETS::GRAPHIC_ = "assets/graphics/"  [extern]
```

Definition at line 4 of file constant.cpp.

6.1.1.3 SOUND_

```
const std::string PASSETS::SOUND_ = "assets/sounds/"  [extern]
```

Definition at line 5 of file constant.cpp.

6.2 PATB Namespace Reference

Variables

- const std::string ATB_ = "atb/"
- const std::string WINDOW_ = "atb/window/"
- const std::string INTERFACE_ = "atb/interface/"
- const std::string BUTTON_ = "atb/button/"
- const std::string CONTAINER_ = "atb/container/"
- const std::string OBJECT_ = "atb/object/"
- const std::string MAP_ = "atb/map/"
- const std::string CHUNK_ = "atb/chunk/"
- const std::string BLOCK_ = "atb/block/"
- const std::string ENTITY_ = "atb/entity/"

6.2.1 Variable Documentation

6.2.1.1 ATB_

```
const std::string PATB::ATB_ = "atb/"  [extern]
```

Definition at line 8 of file constant.cpp.

6.2.1.2 BLOCK_

```
const std::string PATB::BLOCK_ = "atb/block/"  [extern]
```

Definition at line 16 of file constant.cpp.

6.2.1.3 BUTTON_

```
const std::string PATB::BUTTON_ = "atb/button/"  [extern]
```

Definition at line 11 of file constant.cpp.

6.2.1.4 CHUNK_

```
const std::string PATB::CHUNK_ = "atb/chunk/"  [extern]
```

Definition at line 15 of file constant.cpp.

6.2.1.5 CONTAINER_

```
const std::string PATB::CONTAINER_ = "atb/container/"  [extern]
```

Definition at line 12 of file constant.cpp.

6.2.1.6 ENTITY_

```
const std::string PATB::ENTITY_ = "atb/entity/"  [extern]
```

Definition at line 17 of file constant.cpp.

6.2.1.7 INTERFACE_

```
const std::string PATB::INTERFACE_ = "atb/interface/"  [extern]
```

Definition at line 10 of file constant.cpp.

6.2.1.8 MAP_

```
const std::string PATB::MAP_ = "atb/map/"  [extern]
```

Definition at line 14 of file constant.cpp.

6.2.1.9 OBJECT_

```
const std::string PATB::OBJECT_ = "atb/object/"  [extern]
```

Definition at line 13 of file constant.cpp.

6.2.1.10 WINDOW_

```
const std::string PATB::WINDOW_ = "atb/window/"  [extern]
```

Definition at line 9 of file constant.cpp.

6.3 REQUEST Namespace Reference

Enumerations

- enum ID {
  INVALID , NONE , CHANGE_INF , DELAY ,
  LOSE }

6.3.1 Enumeration Type Documentation

6.3.1.1 ID

```
enum REQUEST::ID
```

Enumerator

INVALID	
NONE	
CHANGE_INF	
DELAY	
LOSE	

Definition at line 6 of file request.hpp.

```
    {
        INVALID,
        NONE,
        CHANGE_INF,
        DELAY,
        LOSE,
    };
```

6.4 VECTOR2D Namespace Reference

Functions

- float getAngle (fPoint v1)
- float getAngle (fPoint v1, fPoint v2)

6.4.1 Function Documentation

6.4.1.1 getAngle() [1/2]

```
float VECTOR2D::getAngle (
            fPoint v1 )
```

Definition at line 5 of file vector.cpp.

```
{
    // arctan(y / x)
    return atan2(v1[1], v1[0]);
}
```

6.4.1.2 getAngle() [2/2]

```
float VECTOR2D::getAngle (
            fPoint v1,
            fPoint v2 )
```

Definition at line 16 of file vector.cpp.

```
{
    // angle between 2 vector
    // v1 * v2 = |v1| * |v2| * cos(angle)

    float dot = v1[0] * v2[0] + v1[1] * v2[1];
    float abs1 = sqrt(sqr(v1[0]) + sqr(v1[1]));
    float abs2 = sqrt(sqr(v2[0]) + sqr(v2[1]));
    return acos(dot / (abs1 * abs2));
}
```

6.5 YAML_FILE Namespace Reference

opens and interacts with YAML files

Functions

- bool isFile (std::string path)
- YAML::Node readFile (std::string path)
- bool writeFile (std::string path, YAML::Node content)

6.5.1 Detailed Description

opens and interacts with YAML files

6.5.2 Function Documentation

6.5.2.1 isFile()

```
bool YAML_FILE::isFile (
            std::string path )
```

Definition at line 5 of file file.cpp.

```
{
    // return true if file exists

    std::ifstream fin(path);
    return fin.good();
}
```

6.5.2.2 readFile()

```
YAML::Node YAML_FILE::readFile (
            std::string path )
```

Definition at line 13 of file file.cpp.

```
{
    // return YAML::Node from file

    YAML::Node node;
    try
    {
        node = YAML::LoadFile(path);
    }
    catch (YAML::BadFile& e)
    {
        std::cout « "Error: " « e.what() « std::endl;
    }
    return node;
}
```

6.5.2.3 writeFile()

```
bool YAML_FILE::writeFile (
            std::string path,
            YAML::Node content )
```

Chapter 7

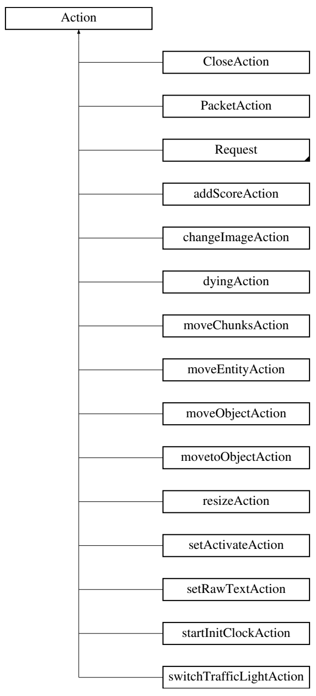
Class Documentation

7.1 Action Class Reference

manages the way an action is executed

`#include <action.hpp>`

Inheritance diagram for Action:



Public Member Functions

- Action ()
- Action (Action $*$)
- virtual $\sim$Action ()=default
- virtual int isRequest ()
- virtual bool isPackage ()
- virtual void execute ()
- virtual Action $*$ clone ()
- virtual std::vector$<$ Action $*$ $>$ unpack ()
- virtual ARGS & getArgs ()

7.1.1 Detailed Description

manages the way an action is executed

Definition at line 31 of file action.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Action() [1/2]

```
Action::Action ( )
```

Definition at line 6 of file action.cpp.

```
00007 {
00008 }
```

7.1.2.2 Action() [2/2]

```
Action::Action (
            Action * action )
```

Definition at line 10 of file action.cpp.

```
00011 {
00012 }
```

7.1.2.3 ∼Action()

```
virtual Action::∼Action ( )  [virtual], [default]
```

7.1.3 Member Function Documentation

7.1.3.1 clone()

```
Action * Action::clone ( )  [virtual]
```

Reimplemented in PacketAction, Request, changeInfRequest, loseRequest, moveEntityAction, changeImageAction, moveChunksAction, and moveObjectAction.

Definition at line 29 of file action.cpp.

```
00030 {
00031     return this;
00032 }
```

7.1.3.2 execute()

```
void Action::execute ( )  [virtual]
```

Reimplemented in CloseAction, resizeAction, PacketAction, moveEntityAction, changeImageAction, moveChunksAction, and moveObjectAction.

Definition at line 25 of file action.cpp.

```
00026 {
00027 }
```

7.1.3.3 getArgs()

`ARGS & Action::getArgs ( ) [virtual]`

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.1.3.4 isPackage()

`bool Action::isPackage ( ) [virtual]`

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
{
    return false;
}
```

7.1.3.5 isRequest()

`int Action::isRequest ( ) [virtual]`

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.

```
{
    return 0;
}
```

7.1.3.6 unpack()

`std::vector< Action * > Action::unpack ( ) [virtual]`

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
{
    return std::vector<Action*> ({this});
}
```

The documentation for this class was generated from the following files:

- src/action/include/action.hpp
- src/action/src/action.cpp

7.2 ARGS Struct Reference

stores request information

`#include <action.hpp>`

Public Member Functions

- ARGS ()=default
- ∼ARGS ()=default
- std::string getInterfaceName ()

Public Attributes

- std::vector< std::string > str
- std::vector< int > num
- std::vector< void ∗ > addr

7.2.1 Detailed Description

stores request information

Definition at line 13 of file action.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ARGS()

```
ARGS::ARGS ( )  [default]
```

7.2.2.2 ∼ARGS()

```
ARGS::∼ARGS ( )  [default]
```

7.2.3 Member Function Documentation

7.2.3.1 getInterfaceName()

```
std::string ARGS::getInterfaceName ( )
```

Definition at line 3 of file args.cpp.

```
{
    return str[0];
}
```

7.2.4 Member Data Documentation

7.2.4.1 addr

```
std::vector<void*> ARGS::addr
```

Definition at line 17 of file action.hpp.

7.2.4.2 num

```
std::vector<int> ARGS::num
```

Definition at line 16 of file action.hpp.

7.2.4.3 str

```
std::vector<std::string> ARGS::str
```

Definition at line 15 of file action.hpp.

The documentation for this struct was generated from the following files:

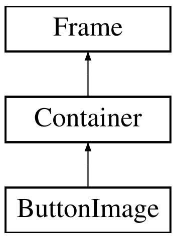
- src/action/include/action.hpp
- src/action/src/args.cpp

7.3 ButtonImage Class Reference

manages the appearance and behavior of a button

```
#include <button.hpp>
```

Inheritance diagram for ButtonImage:



Public Member Functions

- ButtonImage (Frame ∗parrent, Rectangle relative)
- ∼ButtonImage ()
- void draw ()
- PacketAction ∗ react ()
- void changeIndex (int newindex)
- void changePosition (Rectangle change)
- bool isClicked () const
- bool isPressing () const
- int getClicked ()
- std::string linkContent (std::string)
- std::string linkContentAbsolute (std::string)
- std::string getName ()
- void setProbability (int)
- int getProbability ()
- void chooseSprite (int)

choose a specific sprite from a vector of sprites

- void chooseImage (int)

  choose the state of the sprite

- void chooseImage (int, int)

  choose the state of the sprite

- void nextImage ()

  move to next state of the sprite

- void prevImage ()

  move to previous state of the sprite

- void nextSprite ()

  move to the next sprite

- void prevSprite ()

  move to the previous sprite

- bool isOverlapping (fPoint)
- bool isOverlapping (Rectangle)
- bool isOverlapping (Container ∗)
- float OverlappingArea (Rectangle)
- float OverlappingArea (Container ∗)
- void show ()
- void hide ()
- void toggleVisibility ()
- bool isVisible ()
- int getInstanceId ()
- virtual Action ∗ getRuntimeEvent ()
- void plug (Frame ∗par, fRect rel)

  attach a frame to a parent by relative position

- void plug (Frame ∗par)

  attach a frame to a parent by old relative position

- void unplug ()

  detach a frame from its parent

- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (fPoint rel)
- void resize (int w, int h)
- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- void loadEvent (YAML::Node node)
- bool loadName (YAML::Node node)
- void loadSprites (YAML::Node node)
- void loadFocus (YAML::Node node)
- virtual void updateFrame (bool recursive=false)
- bool isroot () const

  return true if this frame is root
- void addSubframe (Frame ∗subframe)

  Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

  Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

7.3.1 Detailed Description

manages the appearance and behavior of a button

Definition at line 18 of file button.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ButtonImage()

```
ButtonImage::ButtonImage (
              Frame ∗ parrent,
              Rectangle relative )
```

Definition at line 5 of file constructor.cpp.

```
00005                                                          : Container(parrent, rel)
00006 {
00007      // set default
00008     this->chooseImage(0, this->tmpPath);
00009     this->color = WHITE;
00010     this->pressing = false;
00011     this->isHover = false;
00012     this->clicked = false;
00013
00014     this->releaseID = -1;
00015     this->hoverID = -1;
00016     this->pressingID = -1;
00017     this->clickedID = -1;
00018 }
```

7.3.2.2 ∼ButtonImage()

```
ButtonImage::∼ButtonImage ( )
```

Definition at line 3 of file destructor.cpp.

```
00004 {
00005     for(auto &action : actions)
00006     {
00007         delete action;
00008     }
00009 }
```

7.3.3 Member Function Documentation

7.3.3.1 addSubframe()

```
void Frame::addSubframe (
            Frame * subframe )  [protected], [inherited]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.

```
{
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}
```

7.3.3.2 beginUpdate()

```
void Frame::beginUpdate ( )  [protected], [inherited]
```

Definition at line 113 of file family.cpp.

```
{
    mtx.lock();
}
```

7.3.3.3 changeIndex()

```
void ButtonImage::changeIndex (
            int newindex )
```

Definition at line 54 of file arthmetic.cpp.

```
{
    tmpPath = newindex;
}
```

7.3.3.4 changePosition()

```
void ButtonImage::changePosition (
            Rectangle change )
```

7.3.3.5 chooseImage() [1/2]

```
void Container::chooseImage (
            int index )  [inherited]
```

choose the state of the sprite

Definition at line 231 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[1] = index;
}
```

7.3.3.6 chooseImage() [2/2]

```
void Container::chooseImage (
              int index,
              int index2 )  [inherited]
```

choose the state of the sprite

Definition at line 238 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    if(index2 < 0 || index2 >= sprites.at(index).size()) return;
    focus[0] = index;
    focus[1] = index2;
}
```

7.3.3.7 chooseSprite()

```
void Container::chooseSprite (
              int index )  [inherited]
```

choose a specific sprite from a vector of sprites

Definition at line 224 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[0] = index;
}
```

7.3.3.8 draw()

```
void ButtonImage::draw ( )  [virtual]
```

Reimplemented from Container.

Definition at line 7 of file arthmetic.cpp.

```
                   {
    if(!isVisible()) return;
    this->Container::draw();
}
```

7.3.3.9 endUpdate()

```
void Frame::endUpdate ( )  [protected], [inherited]
```

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.3.3.10 getCenter()

```
const fPoint & Frame::getCenter ( ) const  [inherited]
```

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.3.3.11 getClicked()

```
int ButtonImage::getClicked ( )
```

Definition at line 59 of file arthmetic.cpp.

```
{
    return tmpPath;
}
```

7.3.3.12 getFrame()

```
const Rectangle & Frame::getFrame ( ) const  [inherited]
```

Definition at line 105 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.3.3.13 getH()

```
const float & Frame::getH ( ) const  [inherited]
```

Definition at line 161 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}
```

7.3.3.14 getInstanceId()

```
int Container::getInstanceId ( )  [inherited]
```

Definition at line 31 of file arthmetic.cpp.

```
{
    return instance_id;
}
```

7.3.3.15 getName()

```
std::string Container::getName ( )  [inherited]
```

Definition at line 275 of file constructor.cpp.

```
{
    return name;
}
```

7.3.3.16 getParent()

```
Frame * Frame::getParent ( )  [inherited]
```

Definition at line 117 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent;
}
```

7.3.3.17 getProbability()

```
int Container::getProbability ( )  [inherited]
```

Definition at line 285 of file constructor.cpp.

```
{
    return probability;
}
```

7.3.3.18 getRelative()

```
const fRect & Frame::getRelative ( ) const  [inherited]
```

Definition at line 111 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.3.3.19 getRuntimeEvent()

```
Action * Container::getRuntimeEvent ( )  [virtual], [inherited]
```

Reimplemented in Chunk, Game, and Interface.

Definition at line 41 of file arthmetic.cpp.

```
{
    return nullptr;
}
```

7.3.3.20 getW()

```
const float & Frame::getW ( ) const  [inherited]
```

Definition at line 155 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}
```

7.3.3.21 getX()

```
const float & Frame::getX ( ) const  [inherited]
```

Definition at line 143 of file arthmetic.cpp.

```
00144 {
00145     std::lock_guard<std::mutex> lock(mtx);
00146     return frame.x;
00147 }
```

7.3.3.22 getY()

```
const float & Frame::getY ( ) const  [inherited]
```

Definition at line 149 of file arthmetic.cpp.

```
00150 {
00151     std::lock_guard<std::mutex> lock(mtx);
00152     return frame.y;
00153 }
```

7.3.3.23 hide()

```
void Container::hide ( )  [inherited]
```

Definition at line 16 of file arthmetic.cpp.

```
00017 {
00018     visible = false;
00019 }
```

7.3.3.24 isClicked()

```
bool ButtonImage::isClicked ( ) const
```

Definition at line 64 of file arthmetic.cpp.

```
00064                                  {
00065     return this->clicked;
00066 }
```

7.3.3.25 isOverlapping() `[1/3]`

```
bool Container::isOverlapping (
            Container * container )  [inherited]
```

Definition at line 16 of file overlap.cpp.

```
00017 {
00018     Rectangle rec = getFrame();
00019     Rectangle rec2 = container->getFrame();
00020     return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
      rec2.height && rec.y + rec.height >= rec2.y);
00021 }
```

7.3.3.26 isOverlapping() [2/3]

```
bool Container::isOverlapping (
             fPoint point )  [inherited]
```

Definition at line 3 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    return (point[0] >= rec.x && point[0] <= rec.x + rec.width && point[1] >= rec.y && point[1] <=
rec.y + rec.height);

}
```

7.3.3.27 isOverlapping() [3/3]

```
bool Container::isOverlapping (
             Rectangle rec )  [inherited]
```

Definition at line 10 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.3.3.28 isPressing()

```
bool ButtonImage::isPressing ( ) const
```

7.3.3.29 isroot()

```
bool Frame::isroot ( ) const  [protected], [inherited]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.3.3.30 isVisible()

```
bool Container::isVisible ( )  [inherited]
```

Definition at line 26 of file arthmetic.cpp.

```
{
    return visible;
}
```

7.3.3.31 linkContent()

```
std::string ButtonImage::linkContent (
             std::string path )  [virtual]
```

Reimplemented from Container.

Definition at line 20 of file constructor.cpp.

```
{
    return linkContentAbsolute(PATB::BUTTON_ + path);
}
```

7.3.3.32 linkContentAbsolute()

```
std::string ButtonImage::linkContentAbsolute (
             std::string path )  [virtual]
```

Reimplemented from Container.

Definition at line 25 of file constructor.cpp.

```
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
    {
        loadSprites(node["textures"]);
        chooseImage(0, 0);
    }
    if(node["events"])
    {
        loadEvent(node["events"]);
    }

    return getName();
}
```

7.3.3.33 loadEvent()

```
void ButtonImage::loadEvent (
             YAML::Node node )  [protected]
```

Definition at line 43 of file constructor.cpp.

```
{
    if(node["hover"])
    {

        for(auto sprite : node["hover"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));
        }
        this->hoverID = actions.size() - 1;
    }

    if(node["release"])
    {
        for(auto sprite : node["release"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));

        }
        this->releaseID = actions.size() - 1;
    }

    if(node["clicked"])
    {
        for(auto sprite : node["clicked"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));
        }
        this->clickedID = actions.size() - 1;
    }

    if(node["pressing"])
    {
        for(auto sprite : node["pressing"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));
        }
        this->pressingID = actions.size() - 1;
    }
}
```

7.3.3.34 loadFocus()

```
void Container::loadFocus (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 218 of file constructor.cpp.

```
{
    focus[0] = node[0].as<int>();
    focus[1] = node[1].as<int>();
}
```

7.3.3.35 loadName()

```
bool Container::loadName (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 111 of file constructor.cpp.

```
{
    if(!node["name"])
    {
        name = "";
        return false;
    }
    name = node["name"].as<std::string>();
    return true;
}
```

7.3.3.36 loadSprites()

```
void Container::loadSprites (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 122 of file constructor.cpp.

```
{
    for(auto sprite : node)
    {
        if(!sprite["path"]) continue;
        if(!sprite["graphics"]) continue;

        std::string path = PASSETS::GRAPHIC_ + sprite["path"].as<std::string>();
        Image image = LoadImage(path.c_str());

        if(sprite["resize"])
        {
            int x = image.width * sprite["resize"][0].as<float>();
            int y = image.height * sprite["resize"][1].as<float>();
            ImageResize(&image, x, y);
        }

        sprites.emplace_back();
        for(auto img : sprite["graphics"])
        {
            float x, y, w, h;
            int repeat = 1;
            int gapX = 0;
            int gapY = 0;

            int dx = 1;
            int dy = 1;

            if(img["x"])
                x = img["x"].as<float>() / 100.0;
            else x = 0;
            if(img["y"])
                y = img["y"].as<float>() / 100.0;
            else y = 0;
            if(img["w"])
                w = img["w"].as<float>() / 100.0;
            else w = 1;
            if(img["h"])
                h = img["h"].as<float>() / 100.0;
            else h = 1;
            if(img["repeat"])
                repeat = img["repeat"].as<int>();
            if(img["gapX"])
                gapX = img["gapX"].as<int>();
            if(img["gapY"])
                gapY = img["gapY"].as<int>();

            if(img["dx"])
                dx = img["dx"].as<int>();
            if(dx < 0) dx = -1;
            else dx = 1;

            if(img["dy"])
                dy = img["dy"].as<int>();
            if(dy < 0) dy = -1;
            else dy = 1;

            int imgw = image.width;
            int imgh = image.height;

            if(img["axis"] && img["axis"].as<std::string>() == "horizontal")
            {
                for(float j = y; j >= 0 && j + h < 1 + 1e-2; j += dy * (gapY + h))
                {
                    for(float i = x; i >= 0 && i + w <= 1 + 1e-2 && repeat--; i += dx * (gapX + w))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }else
            {
                for(float i = x; i >= 0 && i + w <= 1 + 1e-2; i += dx * (gapX + w))
                {
                    for(float j = y; j >= 0 && j + h < 1 + 1e-2 && repeat--; j += dy * (gapY + h))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }
        }
        UnloadImage(image);
    }
}
```

7.3.3.37 moveBy() [1/2]

```
void Frame::moveBy (
            fPoint rel )  [inherited]
```

Definition at line 65 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.38 moveBy() [2/2]

```
void Frame::moveBy (
            int x,
            int y )  [inherited]
```

Definition at line 75 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.39 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
            fPoint rel )  [inherited]
```

Definition at line 43 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.40 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
            int x,
            int y )  [inherited]
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.41 moveTo() [1/2]

```
void Frame::moveTo (
            fPoint rel )  [inherited]
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.42 moveTo() [2/2]

```
void Frame::moveTo (
            int x,
            int y )  [inherited]
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.43 nextImage()

```
void Container::nextImage ( )  [inherited]
```

move to next state of the sprite

Definition at line 247 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]++;
    if(focus[1] >= sprites.at(focus[0]).size()) focus[1] = 0;
}
```

7.3.3.44 nextSprite()

```
void Container::nextSprite ( )  [inherited]
```

move to the next sprite

Definition at line 261 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]++;
    if(focus[0] >= sprites.size()) focus[0] = 0;
}
```

7.3.3.45 operator fRect()

```
Frame::operator fRect ( ) const  [inherited]
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.3.3.46 operator iRect()

```
Frame::operator iRect ( ) const  [inherited]
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.3.3.47 operator Rectangle()

```
Frame::operator Rectangle ( ) const  [inherited]
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.3.3.48 OverlappingArea() [1/2]

```
float Container::OverlappingArea (
            Container * container )  [inherited]
```

Definition at line 34 of file overlap.cpp.

```
{
    Rectangle rec = container->getFrame();
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.3.3.49 OverlappingArea() [2/2]

```
float Container::OverlappingArea (
            Rectangle rec )  [inherited]
```

Definition at line 23 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.3.3.50 plug() [1/2]

```
void Frame::plug (
            Frame * par )  [inherited]
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.3.3.51 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )  [inherited]
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.3.3.52 prevImage()

```
void Container::prevImage ( )  [inherited]
```

move to previous state of the sprite

Definition at line 254 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]--;
    if(focus[1] < 0) focus[1] = sprites.at(focus[0]).size() - 1;
}
```

7.3.3.53 prevSprite()

```
void Container::prevSprite ( )  [inherited]
```

move to the previous sprite

Definition at line 268 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]--;
    if(focus[0] < 0) focus[0] = sprites.size() - 1;
}
```

7.3.3.54 react()

```
PacketAction * ButtonImage::react ( )  [virtual]
```

Reimplemented from Container.

Definition at line 12 of file arthmetic.cpp.

```
                                    {

    if (CheckCollisionPointRec(GetMousePosition(), rectangle)) {
        this->isHover = 1;
        if (IsMouseButtonDown(MOUSE_LEFT_BUTTON)) { // click -> pressing
            this->clicked = true;
            if(this->pressingID == -1)
                return nullptr;
            PacketAction* packet = new PacketAction();
            packet->addAction(actions[pressingID]->clone());
            return packet;
        }
        else if(this->clicked) { // release -> click

            this->clicked = false;
            if(this->clickedID == -1)
                return nullptr;
            PacketAction* packet = new PacketAction();
            packet->addAction(actions[clickedID]->clone());
            packet->addAction(new changeInfRequest("test"));
            return packet;
        }
        if(this->hoverID == -1)
            return nullptr;
        PacketAction* packet = new PacketAction();
            packet->addAction(actions[hoverID]->clone());
            return packet;
    }
    if (this->isHover == 1)
    {
        this->isHover = 0;
        if(this->releaseID == -1)
            return nullptr;
        PacketAction* packet = new PacketAction();
            packet->addAction(actions[releaseID]->clone());
            return packet;
    }
    return nullptr;
}
```

7.3.3.55 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe )  [protected], [inherited]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.

```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.3.3.56 resize() [1/2]

```
void Frame::resize (
             fPoint rel )  [inherited]
```

Definition at line 85 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.57 resize() [2/2]

```
void Frame::resize (
             int w,
             int h )  [inherited]
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.58 setProbability()

```
void Container::setProbability (
             int prob )  [inherited]
```

Definition at line 280 of file constructor.cpp.

```
{
    probability = prob;
}
```

7.3.3.59 setRelative()

```
void Frame::setRelative (
            fRect rel )  [inherited]
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.3.3.60 show()

```
void Container::show ( )  [inherited]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    visible = true;
}
```

7.3.3.61 toggleVisibility()

```
void Container::toggleVisibility ( )  [inherited]
```

Definition at line 21 of file arthmetic.cpp.

```
{
    visible = !visible;
}
```

7.3.3.62 unplug()

```
void Frame::unplug ( )  [inherited]
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.3.3.63 updateFrame()

```
void Frame::updateFrame (
            bool recursive = false )  [protected], [virtual], [inherited]
```

Reimplemented in Visual.

Definition at line 3 of file arthmetic.cpp.

```
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}
```

The documentation for this class was generated from the following files:

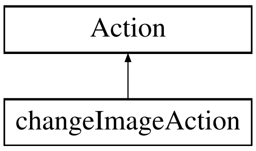
- src/button/include/button.hpp
- src/button/src/arthmetic.cpp
- src/button/src/constructor.cpp
- src/button/src/destructor.cpp

7.4 changeImageAction Class Reference

changes display image of container

`#include <container.hpp>`

Inheritance diagram for changeImageAction:



Public Member Functions

- changeImageAction (Container ∗, iPoint)
- changeImageAction (changeImageAction ∗)
- ∼changeImageAction ()
- void execute () override
- Action ∗ clone () override
- virtual int isRequest ()
- virtual bool isPackage ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

7.4.1 Detailed Description

changes display image of container

Definition at line 108 of file container.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 changeImageAction() [1/2]

```
changeImageAction::changeImageAction (
            Container * c,
            iPoint p )
```

Definition at line 3 of file changesprite.cpp.

```
00004 {
00005     container = c;
00006     focus = p;
00007 }
```

7.4.2.2 changeImageAction() [2/2]

```
changeImageAction::changeImageAction (
            changeImageAction * c )
```

Definition at line 9 of file changesprite.cpp.

```
00010 {
00011     container = c->container;
00012     focus = c->focus;
00013 }
```

7.4.2.3 ∼changeImageAction()

```
changeImageAction::∼changeImageAction ( )
```

Definition at line 15 of file changesprite.cpp.

```
00016 {
00017     container = nullptr;
00018 }
```

7.4.3 Member Function Documentation

7.4.3.1 clone()

```
Action * changeImageAction::clone ( ) [override], [virtual]
```

Reimplemented from Action.

Definition at line 25 of file changesprite.cpp.

```
00026 {
00027     return new changeImageAction(this);
00028 }
```

7.4.3.2 execute()

```
void changeImageAction::execute ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 20 of file changesprite.cpp.

```
{
    container->chooseImage(focus[0], focus[1]);
}
```

7.4.3.3 getArgs()

```
ARGS & Action::getArgs ( )  [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.4.3.4 isPackage()

```
bool Action::isPackage ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
{
    return false;
}
```

7.4.3.5 isRequest()

```
int Action::isRequest ( )  [virtual], [inherited]
```

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.

```
{
    return 0;
}
```

7.4.3.6 unpack()

```
std::vector< Action * > Action::unpack ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
{
    return std::vector<Action*> ({this});
}
```

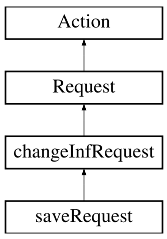
The documentation for this class was generated from the following files:

- src/container/include/container.hpp
- src/container/src/action/changesprite.cpp

7.5 changeInfRequest Class Reference

```
#include <request.hpp>
```

Inheritance diagram for changeInfRequest:



Public Member Functions

- changeInfRequest (std::string s)
- changeInfRequest (changeInfRequest ∗)
- ∼changeInfRequest ()=default
- int isRequest () override
- Action ∗ clone () override
- ARGS & getArgs () override
- virtual bool isPackage ()
- virtual void execute ()
- virtual std::vector< Action ∗ > unpack ()

Protected Attributes

- ARGS args

7.5.1 Detailed Description

Definition at line 27 of file request.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 changeInfRequest() [1/2]

```
changeInfRequest::changeInfRequest (
            std::string s )
```

Definition at line 3 of file changeinf.cpp.

```
00004 {
00005     args.str.push_back(s);
00006 }
```

7.5.2.2 changeInfRequest() [2/2]

```
changeInfRequest::changeInfRequest (
            changeInfRequest * other )
```

Definition at line 8 of file changeinf.cpp.

```
00009 {
00010     args = other->args;
00011 }
```

7.5.2.3 ∼changeInfRequest()

```
changeInfRequest::∼changeInfRequest ( )  [default]
```

7.5.3 Member Function Documentation

7.5.3.1 clone()

```
Action * changeInfRequest::clone ( )  [override], [virtual]
```

Reimplemented from Request.

Definition at line 18 of file changeinf.cpp.

```
00019 {
00020     return new changeInfRequest(this);
00021 }
```

7.5.3.2 execute()

```
void Action::execute ( )  [virtual], [inherited]
```

Reimplemented in CloseAction, resizeAction, PacketAction, moveEntityAction, changeImageAction, moveChunksAction, and moveObjectAction.

Definition at line 25 of file action.cpp.

```
00026 {
00027 }
```

7.5.3.3 getArgs()

```
ARGS & changeInfRequest::getArgs ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 23 of file changeinf.cpp.

```
00024 {
00025     return args;
00026 }
```

7.5.3.4 isPackage()

```
bool Action::isPackage ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
{
    return false;
}
```

7.5.3.5 isRequest()

```
int changeInfRequest::isRequest ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 13 of file changeinf.cpp.

```
{
    return REQUEST::CHANGE_INF;
}
```

7.5.3.6 unpack()

```
std::vector< Action * > Action::unpack ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
{
    return std::vector<Action*> ({this});
}
```

7.5.4 Member Data Documentation

7.5.4.1 args

```
ARGS Request::args  [protected], [inherited]
```

Definition at line 17 of file request.hpp.

The documentation for this class was generated from the following files:

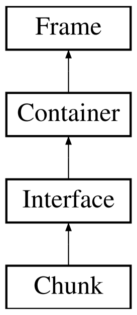
- src/action/include/request.hpp
- src/action/src/request/changeinf.cpp

7.6 Chunk Class Reference

manages the spawning of chunks and how entities interact witht them

```
#include <chunk.hpp>
```

Inheritance diagram for Chunk:



Public Member Functions

- Chunk (Frame ∗, Rectangle)
- Chunk (Chunk ∗)
- Chunk (Chunk ∗, Rectangle)
- Chunk (Chunk ∗, Frame ∗, Rectangle)
- ∼Chunk ()
- void addVisiter (Container ∗)
- void addVisiter (Container ∗, int)
- void addVisiter (Container ∗, Rectangle)
- void addVisiter (Container ∗, int, Rectangle)
- void generateEntity ()
- void setVelocity (fPoint)
- std::string linkContent (std::string path) override
- Action ∗ getRuntimeEvent () override
- void draw () override
- Container ∗ getContainers (int)
- int getContainersSize ()
- std::string linkContentAbsolute (std::string path) override
- Action ∗ react () override
- std::string getName ()
- void setProbability (int)
- int getProbability ()
- void chooseSprite (int)

  choose a specific sprite from a vector of sprites
- void chooseImage (int)

  choose the state of the sprite
- void chooseImage (int, int)

  choose the state of the sprite
- void nextImage ()

  move to next state of the sprite
- void prevImage ()

  move to previous state of the sprite
- void nextSprite ()

move to the next sprite
- void prevSprite ()

  move to the previous sprite
- bool isOverlapping (fPoint)
- bool isOverlapping (Rectangle)
- bool isOverlapping (Container ∗)
- float OverlappingArea (Rectangle)
- float OverlappingArea (Container ∗)
- void show ()
- void hide ()
- void toggleVisibility ()
- bool isVisible ()
- int getInstanceId ()
- void plug (Frame ∗par, fRect rel)

  attach a frame to a parent by relative position
- void plug (Frame ∗par)

  attach a frame to a parent by old relative position
- void unplug ()

  detach a frame from its parent
- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (fPoint rel)
- void resize (int w, int h)
- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- void drawEntity ()
- Container ∗ randomEntity ()
- void movingEntity ()
- void loadObject (YAML::Node)
- void loadControl (YAML::Node)
- void loadButton (YAML::Node)
- void drawNested ()
- void drawContainers ()
- bool loadName (YAML::Node node)
- void loadSprites (YAML::Node node)
- void loadFocus (YAML::Node node)

- virtual void updateFrame (bool recursive=false)
- bool isroot () const

  return true if this frame is root
- void addSubframe (Frame ∗subframe)

  Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

  Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

Friends

- class moveEntityAction

7.6.1 Detailed Description

manages the spawning of chunks and how entities interact witht them

Definition at line 16 of file chunk.hpp.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Chunk() [1/4]

```
Chunk::Chunk (
              Frame ∗ frame,
              Rectangle rect )
```

Definition at line 4 of file constructor.cpp.

```
                                          : Interface(frame, rect)
{

}
```

7.6.2.2 Chunk() [2/4]

```
Chunk::Chunk (
              Chunk ∗ other )
```

Definition at line 9 of file constructor.cpp.

```
                           : Interface(other)
{

    for(auto i : other->visiter)
    {
        Rectangle rel;
        rel.x = 1;
        rel.y = -0.375;
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        visiter.push_back(new Container(i, this, rel));
    }
    velocity = other->velocity;
    generateEntity();
}
```

7.6.2.3 Chunk() [3/4]

```
Chunk::Chunk (
            Chunk * other,
            Rectangle rect )
```

Definition at line 25 of file constructor.cpp.

```
                                                : Interface(other, rect)
{
    for(auto i : other->visiter)
    {
        Rectangle rel;
        rel.x = 1;
        rel.y = -0.375;
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        visiter.push_back(new Container(i, this, rel));
    }
    velocity = other->velocity;
    generateEntity();
}
```

7.6.2.4 Chunk() [4/4]

```
Chunk::Chunk (
            Chunk * other,
            Frame * frame,
            Rectangle rect )
```

Definition at line 40 of file constructor.cpp.

```
                                                              : Interface(other, frame, rect)
{
    for(auto i : other->visiter)
    {
        Rectangle rel;
        rel.x = 1;
        rel.y = -0.375;
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        visiter.push_back(new Container(i, this, rel));
    }
    velocity = other->velocity;
    generateEntity();
}
```

7.6.2.5 ∼Chunk()

```
Chunk::∼Chunk ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    for(auto i : visiter)
        delete i;

    while(!Entity.empty())
    {
        delete Entity.back();
        Entity.pop_back();
    }
}
```

7.6.3 Member Function Documentation

7.6.3.1 addSubframe()

```
void Frame::addSubframe (
            Frame * subframe )   [protected], [inherited]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.

```
{
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}
```

7.6.3.2 addVisiter() [1/4]

```
void Chunk::addVisiter (
            Container * obj )
```

Definition at line 82 of file constructor.cpp.

```
{
    Rectangle rel;
    rel.x = obj->getRelative()[0];
    rel.y = obj->getRelative()[1];
    rel.width = obj->getRelative()[2];
    rel.height = obj->getRelative()[3];

    Container* c = new Container(obj, this, rel);
    visiter.push_back(c);
}
```

7.6.3.3 addVisiter() [2/4]

```
void Chunk::addVisiter (
            Container * obj,
            int prob )
```

Definition at line 94 of file constructor.cpp.

```
{
    Rectangle rel;
    rel.x = obj->getRelative()[0];
    rel.y = obj->getRelative()[1];
    rel.width = obj->getRelative()[2];
    rel.height = obj->getRelative()[3];

    Container* c = new Container(obj, this, rel);
    c->setProbability(prob);
    visiter.push_back(c);
}
```

7.6.3.4 addVisiter() [3/4]

```
void Chunk::addVisiter (
            Container * obj,
            int prob,
            Rectangle rel )
```

Definition at line 113 of file constructor.cpp.

```
{
    Container* c = new Container(obj, this, rel);
    c->setProbability(prob);
    visiter.push_back(c);
}
```

7.6.3.5 addVisiter() [4/4]

```
void Chunk::addVisiter (
              Container * obj,
              Rectangle rel )
```

Definition at line 107 of file constructor.cpp.

```
{
    Container* c = new Container(obj, this, rel);
    visiter.push_back(c);
}
```

7.6.3.6 beginUpdate()

```
void Frame::beginUpdate ( )  [protected], [inherited]
```

Definition at line 113 of file family.cpp.

```
{
    mtx.lock();
}
```

7.6.3.7 chooseImage() [1/2]

```
void Container::chooseImage (
              int index )  [inherited]
```

choose the state of the sprite

Definition at line 231 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[1] = index;
}
```

7.6.3.8 chooseImage() [2/2]

```
void Container::chooseImage (
              int index,
              int index2 )  [inherited]
```

choose the state of the sprite

Definition at line 238 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    if(index2 < 0 || index2 >= sprites.at(index).size()) return;
    focus[0] = index;
    focus[1] = index2;
}
```

7.6.3.9 chooseSprite()

```
void Container::chooseSprite (
              int index )  [inherited]
```

choose a specific sprite from a vector of sprites

Definition at line 224 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[0] = index;
}
```

7.6.3.10 draw()

```
void Chunk::draw ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 9 of file arthmetic.cpp.

```
{

    Container::draw();
    drawNested();
    drawContainers();
    drawEntity();
}
```

7.6.3.11 drawContainers()

```
void Interface::drawContainers ( )  [protected], [inherited]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    for(auto& child : containers)
    {
        child->draw();
    }
}
```

7.6.3.12 drawEntity()

```
void Chunk::drawEntity ( )  [protected]
```

Definition at line 3 of file arthmetic.cpp.

```
{
    for(auto i : Entity)
        i->draw();
}
```

7.6.3.13 drawNested()

```
void Interface::drawNested ( )  [protected], [inherited]
```

Definition at line 3 of file arthmetic.cpp.

```
{
    for(auto& child : nested)
    {
        child->draw();
    }
}
```

7.6.3.14 endUpdate()

```
void Frame::endUpdate ( )  [protected], [inherited]
```

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.6.3.15 generateEntity()

```
void Chunk::generateEntity ( )
```

Definition at line 55 of file constructor.cpp.

```
{

    if(visiter.empty()) return;
    float x = GetRandomValue(-40, 10);

    while(x < 0.9)
    {
        Container* c = randomEntity();
        Rectangle rel;
        rel.x = x;
        rel.y = -0.375;
        rel.width = c->getRelative()[2];
        rel.height = c->getRelative()[3];
        Container* cont = new Container(c, this, rel);
        Entity.push_back(cont);
        x += GetRandomValue(20, 60) / 100.0;
    }
}
```

7.6.3.16 getCenter()

```
const fPoint & Frame::getCenter ( ) const  [inherited]
```

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.6.3.17 getContainers()

```
Container * Interface::getContainers (
              int id )  [inherited]
```

Definition at line 176 of file constructor.cpp.

```
{
    if(id < 0 || id >= containers.size()) return nullptr;
    return containers[id];
}
```

7.6.3.18 getContainersSize()

```
int Interface::getContainersSize ( )  [inherited]
```

Definition at line 182 of file constructor.cpp.

```
{
    return containers.size();
}
```

7.6.3.19 getFrame()

```
const Rectangle & Frame::getFrame ( ) const  [inherited]
```

Definition at line 105 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.6.3.20 getH()

```
const float & Frame::getH ( ) const  [inherited]
```

Definition at line 161 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}
```

7.6.3.21 getInstanceId()

```
int Container::getInstanceId ( )  [inherited]
```

Definition at line 31 of file arthmetic.cpp.

```
{
    return instance_id;
}
```

7.6.3.22 getName()

```
std::string Container::getName ( )  [inherited]
```

Definition at line 275 of file constructor.cpp.

```
{
    return name;
}
```

7.6.3.23 getParent()

```
Frame * Frame::getParent ( )  [inherited]
```

Definition at line 117 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent;
}
```

7.6.3.24 getProbability()

```
int Container::getProbability ( )  [inherited]
```

Definition at line 285 of file constructor.cpp.

```
{
    return probability;
}
```

7.6.3.25 getRelative()

```
const fRect & Frame::getRelative ( ) const  [inherited]
```

Definition at line 111 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.6.3.26 getRuntimeEvent()

```
Action * Chunk::getRuntimeEvent ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 26 of file action.cpp.

```
{
    PacketAction* packet = nullptr;
    Action* action = Interface::getRuntimeEvent();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    if(std::chrono::system_clock::now() - moveClock >= moveTime)
    {
        Action* action = new moveEntityAction(this);
        if(packet == nullptr)
        {
            packet = new PacketAction();
        }
        packet->addAction(action);
        moveClock = std::chrono::system_clock::now();
    }
    return packet;
}
```

7.6.3.27 getW()

```
const float & Frame::getW ( ) const  [inherited]
```

Definition at line 155 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}
```

7.6.3.28 getX()

```
const float & Frame::getX ( ) const  [inherited]
```

Definition at line 143 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}
```

7.6.3.29 getY()

```
const float & Frame::getY ( ) const  [inherited]
```

Definition at line 149 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}
```

7.6.3.30 hide()

```
void Container::hide ( )  [inherited]
```

Definition at line 16 of file arthmetic.cpp.

```
{
    visible = false;
}
```

7.6.3.31 isOverlapping() [1/3]

```
bool Container::isOverlapping (
            Container * container )  [inherited]
```

Definition at line 16 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    Rectangle rec2 = container->getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
      rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.6.3.32 isOverlapping() [2/3]

```
bool Container::isOverlapping (
            fPoint point )  [inherited]
```

Definition at line 3 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    return (point[0] >= rec.x && point[0] <= rec.x + rec.width && point[1] >= rec.y && point[1] <=
      rec.y + rec.height);

}
```

7.6.3.33 isOverlapping() [3/3]

```
bool Container::isOverlapping (
            Rectangle rec )  [inherited]
```

Definition at line 10 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
      rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.6.3.34 isroot()

```
bool Frame::isroot ( ) const  [protected], [inherited]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.6.3.35 isVisible()

```
bool Container::isVisible ( )  [inherited]
```

Definition at line 26 of file arthmetic.cpp.

```
{
    return visible;
}
```

7.6.3.36 linkContent()

```
std::string Chunk::linkContent (
              std::string path )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 76 of file constructor.cpp.

```
{
    return linkContentAbsolute(PATB::CHUNK_ + path);
}
```

7.6.3.37 linkContentAbsolute()

```
std::string Interface::linkContentAbsolute (
              std::string path )  [override], [virtual], [inherited]
```

Reimplemented from Container.

Definition at line 83 of file constructor.cpp.

```
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
        loadSprites(node["textures"]);

    if(node["focus"])
        loadFocus(node["focus"]);
    else chooseImage(0, 0);

    if(node["object"])
        loadObject(node["object"]);

    if(node["control"])
        loadControl(node["control"]);

    if(node["button"])
        loadButton(node["button"]);

//    if(node["collide"])
//        loadCollide(node["collide"]);

//    if(node["chunk"])
//        loadChunk(node["chunk"]);


//    if(node["event"])
//        loadEvent(node["event"]);

    return getName();
}
```

7.6.3.38 loadButton()

```
void Interface::loadButton (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 159 of file constructor.cpp.

```
{
    for(auto i : node)
    {
        Rectangle rel({0, 0, 0, 0});
        if(i["x"]) rel.x = i["x"].as<float>() / 100;
        if(i["y"]) rel.y = i["y"].as<float>() / 100;
        if(i["w"]) rel.width = i["w"].as<float>() / 100;
        if(i["h"]) rel.height = i["h"].as<float>() / 100;
        ButtonImage *obj;
        obj = new ButtonImage(this, rel);
        obj->linkContent(i["path"].as<std::string>());
        obj->show();
        containers.push_back(obj);
    }
}
```

7.6.3.39 loadControl()

```
void Interface::loadControl (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 134 of file constructor.cpp.

```
{
    for(auto stroke : node)
    {
        KeyStroke* k = new KeyStroke();
        for(auto key : stroke["key"])
        {
            k->add(toKey(key.as<std::string>()));
        }
        std::string action = stroke["action"].as<std::string>();

        if(action == "move-object")
        {
            int id = stroke["args"][0].as<int>();
            float v = stroke["args"][1].as<float>() / 100.0;
            float x = stroke["args"][2].as<float>();
            float y = stroke["args"][3].as<float>();
            moveObjectAction* action = new moveObjectAction(containers[id], fPoint({x, y}), v);
            k->addAction(action);
        }

        keystrokes.push_back(k);
    }
}
```

7.6.3.40 loadFocus()

```
void Container::loadFocus (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 218 of file constructor.cpp.

```
{
    focus[0] = node[0].as<int>();
    focus[1] = node[1].as<int>();
}
```

7.6.3.41 loadName()

```
bool Container::loadName (
                YAML::Node node )  [protected], [inherited]
```

Definition at line 111 of file constructor.cpp.

```
{
    if(!node["name"])
    {
        name = "";
        return false;
    }
    name = node["name"].as<std::string>();
    return true;
}
```

7.6.3.42 loadObject()

```
void Interface::loadObject (
                YAML::Node node )  [protected], [inherited]
```

Definition at line 117 of file constructor.cpp.

```
{
    for(auto i : node)
    {
        Rectangle rel({0, 0, 0, 0});
        if(i["x"]) rel.x = i["x"].as<float>() / 100;
        if(i["y"]) rel.y = i["y"].as<float>() / 100;
        if(i["w"]) rel.width = i["w"].as<float>() / 100;
        if(i["h"]) rel.height = i["h"].as<float>() / 100;
        Container *obj;
        obj = new Object(this, rel);
        obj->linkContent(i["path"].as<std::string>());
        containers.push_back(obj);
    }
}
```

7.6.3.43 loadSprites()

```
void Container::loadSprites (
                YAML::Node node )  [protected], [inherited]
```

Definition at line 122 of file constructor.cpp.

```
{
    for(auto sprite : node)
    {
        if(!sprite["path"]) continue;
        if(!sprite["graphics"]) continue;

        std::string path = PASSETS::GRAPHIC_ + sprite["path"].as<std::string>();
        Image image = LoadImage(path.c_str());

        if(sprite["resize"])
        {
            int x = image.width * sprite["resize"][0].as<float>();
            int y = image.height * sprite["resize"][1].as<float>();
            ImageResize(&image, x, y);
        }

        sprites.emplace_back();
        for(auto img : sprite["graphics"])
        {
            float x, y, w, h;
            int repeat = 1;
            int gapX = 0;
            int gapY = 0;

            int dx = 1;
            int dy = 1;

            if(img["x"])
                x = img["x"].as<float>() / 100.0;
            else x = 0;
            if(img["y"])
                y = img["y"].as<float>() / 100.0;
            else y = 0;
            if(img["w"])
                w = img["w"].as<float>() / 100.0;
            else w = 1;
            if(img["h"])
                h = img["h"].as<float>() / 100.0;
            else h = 1;
            if(img["repeat"])
                repeat = img["repeat"].as<int>();
            if(img["gapX"])
                gapX = img["gapX"].as<int>();
            if(img["gapY"])
                gapY = img["gapY"].as<int>();

            if(img["dx"])
                dx = img["dx"].as<int>();
            if(dx < 0) dx = -1;
            else dx = 1;

            if(img["dy"])
                dy = img["dy"].as<int>();
            if(dy < 0) dy = -1;
            else dy = 1;

            int imgw = image.width;
            int imgh = image.height;

            if(img["axis"] && img["axis"].as<std::string>() == "horizontal")
            {
                for(float j = y; j >= 0 && j + h < 1 + 1e-2; j += dy * (gapY + h))
                {
                    for(float i = x; i >= 0 && i + w <= 1 + 1e-2 && repeat--; i += dx * (gapX + w))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }else
            {
                for(float i = x; i >= 0 && i + w <= 1 + 1e-2; i += dx * (gapX + w))
                {
                    for(float j = y; j >= 0 && j + h < 1 + 1e-2 && repeat--; j += dy * (gapY + h))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }
        }
        UnloadImage(image);
    }
}
```

7.6.3.44 moveBy() [1/2]

```
void Frame::moveBy (
            fPoint rel )  [inherited]
```

Definition at line 65 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.45 moveBy() [2/2]

```
void Frame::moveBy (
            int x,
            int y )  [inherited]
```

Definition at line 75 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.46 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
            fPoint rel )  [inherited]
```

Definition at line 43 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.47 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
            int x,
            int y )  [inherited]
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.48 moveTo() [1/2]

```
void Frame::moveTo (
            fPoint rel )  [inherited]
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.49 moveTo() [2/2]

```
void Frame::moveTo (
            int x,
            int y )  [inherited]
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.50 movingEntity()

```
void Chunk::movingEntity ( )  [protected]
```

Definition at line 17 of file action.cpp.

```
{
    for(auto i : Entity)
    {
        i->moveBy(velocity);
        i->nextImage();
    }
}
```

7.6.3.51 nextImage()

```
void Container::nextImage ( )  [inherited]
```

move to next state of the sprite

Definition at line 247 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]++;
    if(focus[1] >= sprites.at(focus[0]).size()) focus[1] = 0;
}
```

7.6.3.52 nextSprite()

```
void Container::nextSprite ( )  [inherited]
```

move to the next sprite

Definition at line 261 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]++;
    if(focus[0] >= sprites.size()) focus[0] = 0;
}
```

7.6.3.53 operator fRect()

```
Frame::operator fRect ( ) const  [inherited]
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.6.3.54 operator iRect()

```
Frame::operator iRect ( ) const  [inherited]
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.6.3.55 operator Rectangle()

```
Frame::operator Rectangle ( ) const  [inherited]
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.6.3.56 OverlappingArea() [1/2]

```
float Container::OverlappingArea (
            Container * container )  [inherited]
```

Definition at line 34 of file overlap.cpp.

```
{
    Rectangle rec = container->getFrame();
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.6.3.57 OverlappingArea() [2/2]

```
float Container::OverlappingArea (
            Rectangle rec )  [inherited]
```

Definition at line 23 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.6.3.58 plug() [1/2]

```
void Frame::plug (
            Frame * par )  [inherited]
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.6.3.59 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )  [inherited]
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.6.3.60 prevImage()

```
void Container::prevImage ( )  [inherited]
```

move to previous state of the sprite

Definition at line 254 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]--;
    if(focus[1] < 0) focus[1] = sprites.at(focus[0]).size() - 1;
}
```

7.6.3.61 prevSprite()

```
void Container::prevSprite ( )  [inherited]
```

move to the previous sprite

Definition at line 268 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]--;
    if(focus[0] < 0) focus[0] = sprites.size() - 1;
}
```

7.6.3.62 randomEntity()

```
Container * Chunk::randomEntity ( )  [protected]
```

Definition at line 4 of file action.cpp.

```
{
    int value = GetRandomValue(0, 100);

    for(auto i : visiter)
    {
        value -= i->getProbability();
        if(value <= 0) return i;
    }

    return visiter[GetRandomValue(0, visiter.size() - 1)];
}
```

7.6.3.63 react()

```
Action * Interface::react ( )  [override], [virtual], [inherited]
```

Reimplemented from Container.

Definition at line 38 of file action.cpp.

```
{
    if(!isVisible()) return nullptr;
    PacketAction* packet = nullptr;

    Action* action = Container::react();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    for(auto i : keystrokes)
    {
        Action* action = i->react();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    for(auto i : containers)
    {
        Action* action = i->react();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    return packet;
}
```

7.6.3.64 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe )  [protected], [inherited]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.

```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.6.3.65 resize() [1/2]

```
void Frame::resize (
            fPoint rel )  [inherited]
```

Definition at line 85 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.66 resize() [2/2]

```
void Frame::resize (
            int w,
            int h )  [inherited]
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.67 setProbability()

```
void Container::setProbability (
             int prob )  [inherited]
```

Definition at line 280 of file constructor.cpp.

```
{
    probability = prob;
}
```

7.6.3.68 setRelative()

```
void Frame::setRelative (
             fRect rel )  [inherited]
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.6.3.69 setVelocity()

```
void Chunk::setVelocity (
             fPoint vel )
```

Definition at line 120 of file constructor.cpp.

```
{
    velocity = vel;
}
```

7.6.3.70 show()

```
void Container::show ( )  [inherited]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    visible = true;
}
```

7.6.3.71 toggleVisibility()

```
void Container::toggleVisibility ( )  [inherited]
```

Definition at line 21 of file arthmetic.cpp.

```
{
    visible = !visible;
}
```

7.6.3.72 unplug()

```
void Frame::unplug ( )  [inherited]
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.6.3.73 updateFrame()

```
void Frame::updateFrame (
            bool recursive = false )  [protected], [virtual], [inherited]
```

Reimplemented in Visual.

Definition at line 3 of file arthmetic.cpp.

```
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}
```

7.6.4 Friends And Related Symbol Documentation

7.6.4.1 moveEntityAction

```
friend class moveEntityAction  [friend]
```

Definition at line 19 of file chunk.hpp.

The documentation for this class was generated from the following files:

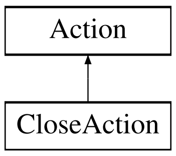
- src/chunk/include/chunk.hpp
- src/chunk/src/action.cpp
- src/chunk/src/arthmetic.cpp
- src/chunk/src/constructor.cpp
- src/chunk/src/destructor.cpp

7.7 CloseAction Class Reference

manages the closing of the application

```
#include <window.hpp>
```

Inheritance diagram for CloseAction:



Public Member Functions

- CloseAction (Window ∗win)
- ∼CloseAction ()=default
- void execute ()
- virtual int isRequest ()
- virtual bool isPackage ()
- virtual Action ∗ clone ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

7.7.1 Detailed Description

manages the closing of the application

Definition at line 181 of file window.hpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 CloseAction()

```
CloseAction::CloseAction (
        Window ∗ win )
```

Definition at line 3 of file close.cpp.

```
00004 {
00005     win = window;
00006 }
```

7.7.2.2 ∼CloseAction()

```
CloseAction::∼CloseAction ( )  [default]
```

7.7.3 Member Function Documentation

7.7.3.1 clone()

```
Action * Action::clone ( )  [virtual], [inherited]
```

Reimplemented in PacketAction, Request, changeInfRequest, loseRequest, moveEntityAction, changeImageAction, moveChunksAction, and moveObjectAction.

Definition at line 29 of file action.cpp.

```
00030 {
00031     return this;
00032 }
```

7.7.3.2 execute()

```
void CloseAction::execute ( )  [virtual]
```

Reimplemented from Action.

Definition at line 8 of file close.cpp.

```
00009 {
00010     win->Wcontent.setStatus(false);
00011 }
```

7.7.3.3 getArgs()

```
ARGS & Action::getArgs ( )  [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
00040 {
00041     return NONE_ARGS;
00042 }
```

7.7.3.4 isPackage()

```
bool Action::isPackage ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
00021 {
00022     return false;
00023 }
```

7.7.3.5 isRequest()

```
int Action::isRequest ( )  [virtual], [inherited]
```

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.

```
00016 {
00017     return 0;
00018 }
```

7.7.3.6 unpack()

```
std::vector< Action * > Action::unpack ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
00035 {
00036     return std::vector<Action*> ({this});
00037 }
```

The documentation for this class was generated from the following files:

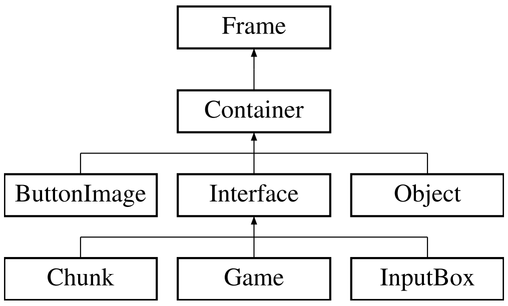
- src/window/include/window.hpp
- src/window/src/action/close.cpp

7.8 Container Class Reference

holds specific entities and their behavior

```
#include <container.hpp>
```

Inheritance diagram for Container:



Public Member Functions

- Container (Frame ∗, Rectangle)
- Container (Container ∗)
- Container (Container ∗, Rectangle)
- Container (Container ∗, Frame ∗, Rectangle)
- virtual ∼Container ()
- virtual std::string linkContent (std::string)
- virtual std::string linkContentAbsolute (std::string)
- std::string getName ()
- void setProbability (int)
- int getProbability ()
- void chooseSprite (int)

    choose a specific sprite from a vector of sprites
- void chooseImage (int)

    choose the state of the sprite
- void chooseImage (int, int)

choose the state of the sprite
- void nextImage ()

    move to next state of the sprite
- void prevImage ()

    move to previous state of the sprite
- void nextSprite ()

    move to the next sprite
- void prevSprite ()

    move to the previous sprite
- bool isOverlapping (fPoint)
- bool isOverlapping (Rectangle)
- bool isOverlapping (Container ∗)
- float OverlappingArea (Rectangle)
- float OverlappingArea (Container ∗)
- virtual void draw ()
- void show ()
- void hide ()
- void toggleVisibility ()
- bool isVisible ()
- int getInstanceId ()
- virtual Action ∗ react ()
- virtual Action ∗ getRuntimeEvent ()
- void plug (Frame ∗par, fRect rel)

    attach a frame to a parent by relative position
- void plug (Frame ∗par)

    attach a frame to a parent by old relative position
- void unplug ()

    detach a frame from its parent
- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (fPoint rel)
- void resize (int w, int h)
- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- bool loadName (YAML::Node node)
- void loadSprites (YAML::Node node)
- void loadFocus (YAML::Node node)
- virtual void updateFrame (bool recursive=false)
- bool isroot () const

  return true if this frame is root
- void addSubframe (Frame ∗subframe)

  Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

  Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

Friends

- class changeImageAction

7.8.1 Detailed Description

holds specific entities and their behavior

Definition at line 19 of file container.hpp.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Container() [1/4]

```
Container::Container (
            Frame * parent,
            Rectangle rect )
```

Definition at line 10 of file constructor.cpp.

```
                                                         : Frame(parent, rect)
{
    instance_id = id_count++;
    focus = {0, 0};
    visible = true;
}
```

7.8.2.2 Container() [2/4]

```
Container::Container (
            Container * other )
```

Definition at line 17 of file constructor.cpp.

```
                                          : Frame(other)
{
    instance_id = id_count++;
    focus = {0, 0};
    name = other->name;
    visible = true;

    for(auto s : other->sprites)
    {
        sprites.emplace_back();
        Rectangle rect;
        rect.x = other->getRelative()[0];
        rect.y = other->getRelative()[1];
        rect.width = other->getRelative()[2];
        rect.height = other->getRelative()[3];

        for(auto v : s)
        {
            sprites.back().push_back(new Visual(v, this, rect));
        }
    }
}
```

7.8.2.3 Container() [3/4]

```
Container::Container (
            Container * other,
            Rectangle rect )
```

Definition at line 40 of file constructor.cpp.

```
                                                         : Frame(other)
{
    instance_id = id_count++;
    focus = {0, 0};
    name = other->name;
    setRelative({rect.x, rect.y, rect.width, rect.height});
    visible = true;

    for(auto s : other->sprites)
    {
        sprites.emplace_back();
        Rectangle rect;
        rect.x = other->getRelative()[0];
        rect.y = other->getRelative()[1];
        rect.width = other->getRelative()[2];
        rect.height = other->getRelative()[3];

        for(auto v : s)
        {
            sprites.back().push_back(new Visual(v, this, rect));
        }
    }
}
```

7.8.2.4 Container() [4/4]

```
Container::Container (
            Container * other,
            Frame * parent,
            Rectangle rect )
```

Definition at line 64 of file constructor.cpp.

```
                                                                          : Frame(parent, rect)
{
    instance_id = id_count++;
    focus = {0, 0};
    name = other->name;
    visible = true;
    for(auto s : other->sprites)
    {
        sprites.emplace_back();
        Rectangle rect;
        rect.x = other->getRelative()[0];
        rect.y = other->getRelative()[1];
        rect.width = other->getRelative()[2];
        rect.height = other->getRelative()[3];

        for(auto v : s)
        {
            sprites.back().push_back(new Visual(v, this, rect));
        }
    }
}
```

7.8.2.5 ∼Container()

`Container::∼Container ( ) [virtual]`

Definition at line 3 of file destructor.cpp.

```
{
    for(Sprite & sprite : sprites)
    {
        for(auto& frame : sprite)
            delete frame;
        sprite.clear();
    }
}
```

7.8.3 Member Function Documentation

7.8.3.1 addSubframe()

```
void Frame::addSubframe (
              Frame * subframe ) [protected], [inherited]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.

```
{
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}
```

7.8.3.2 beginUpdate()

`void Frame::beginUpdate ( ) [protected], [inherited]`

Definition at line 113 of file family.cpp.

```
{
    mtx.lock();
}
```

7.8.3.3 chooseImage() [1/2]

```
void Container::chooseImage (
            int index )
```

choose the state of the sprite

Definition at line 231 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[1] = index;
}
```

7.8.3.4 chooseImage() [2/2]

```
void Container::chooseImage (
            int index,
            int index2 )
```

choose the state of the sprite

Definition at line 238 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    if(index2 < 0 || index2 >= sprites.at(index).size()) return;
    focus[0] = index;
    focus[1] = index2;
}
```

7.8.3.5 chooseSprite()

```
void Container::chooseSprite (
            int index )
```

choose a specific sprite from a vector of sprites

Definition at line 224 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[0] = index;
}
```

7.8.3.6 draw()

```
void Container::draw ( )  [virtual]
```

Reimplemented in ButtonImage, Chunk, Game, Interface, and Object.

Definition at line 4 of file arthmetic.cpp.

```
{
    if(sprites.empty()) return;
    if(!visible) return;
    sprites[focus[0]][focus[1]]->draw();
}
```

7.8.3.7 endUpdate()

```
void Frame::endUpdate ( )  [protected], [inherited]
```

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.8.3.8 getCenter()

```
const fPoint & Frame::getCenter ( ) const  [inherited]
```

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.8.3.9 getFrame()

```
const Rectangle & Frame::getFrame ( ) const  [inherited]
```

Definition at line 105 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.8.3.10 getH()

```
const float & Frame::getH ( ) const  [inherited]
```

Definition at line 161 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}
```

7.8.3.11 getInstanceId()

```
int Container::getInstanceId ( )
```

Definition at line 31 of file arthmetic.cpp.

```
{
    return instance_id;
}
```

7.8.3.12 getName()

```
std::string Container::getName ( )
```

Definition at line 275 of file constructor.cpp.

```
00276 {
00277     return name;
00278 }
```

7.8.3.13 getParent()

```
Frame * Frame::getParent ( )  [inherited]
```

Definition at line 117 of file arthmetic.cpp.

```
00118 {
00119     std::lock_guard<std::mutex> lock(mtx);
00120     return parent;
00121 }
```

7.8.3.14 getProbability()

```
int Container::getProbability ( )
```

Definition at line 285 of file constructor.cpp.

```
00286 {
00287     return probability;
00288 }
```

7.8.3.15 getRelative()

```
const fRect & Frame::getRelative ( ) const  [inherited]
```

Definition at line 111 of file arthmetic.cpp.

```
00112 {
00113     std::lock_guard<std::mutex> lock(mtx);
00114     return relative;
00115 }
```

7.8.3.16 getRuntimeEvent()

```
Action * Container::getRuntimeEvent ( )  [virtual]
```

Reimplemented in Chunk, Game, and Interface.

Definition at line 41 of file arthmetic.cpp.

```
00042 {
00043     return nullptr;
00044 }
```

7.8.3.17 getW()

```
const float & Frame::getW ( ) const  [inherited]
```

Definition at line 155 of file arthmetic.cpp.

```
00156 {
00157     std::lock_guard<std::mutex> lock(mtx);
00158     return frame.width;
00159 }
```

7.8.3.18 getX()

```
const float & Frame::getX ( ) const  [inherited]
```

Definition at line 143 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}
```

7.8.3.19 getY()

```
const float & Frame::getY ( ) const  [inherited]
```

Definition at line 149 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}
```

7.8.3.20 hide()

```
void Container::hide ( )
```

Definition at line 16 of file arthmetic.cpp.

```
{
    visible = false;
}
```

7.8.3.21 isOverlapping() [1/3]

```
bool Container::isOverlapping (
            Container * container )
```

Definition at line 16 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    Rectangle rec2 = container->getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
      rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.8.3.22 isOverlapping() [2/3]

```
bool Container::isOverlapping (
            fPoint point )
```

Definition at line 3 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    return (point[0] >= rec.x && point[0] <= rec.x + rec.width && point[1] >= rec.y && point[1] <=
      rec.y + rec.height);

}
```

7.8.3.23 isOverlapping() [3/3]

```
bool Container::isOverlapping (
            Rectangle rec )
```

Definition at line 10 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
     rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.8.3.24 isroot()

```
bool Frame::isroot ( ) const  [protected], [inherited]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.8.3.25 isVisible()

```
bool Container::isVisible ( )
```

Definition at line 26 of file arthmetic.cpp.

```
{
    return visible;
}
```

7.8.3.26 linkContent()

```
std::string Container::linkContent (
            std::string path )  [virtual]
```

Reimplemented in Chunk, Interface, ButtonImage, and Object.

Definition at line 86 of file constructor.cpp.

```
{
    focus = {0, 0};
    return linkContentAbsolute(PATB::CONTAINER_ + path);
}
```

7.8.3.27 linkContentAbsolute()

```
std::string Container::linkContentAbsolute (
            std::string path )  [virtual]
```

Reimplemented in Game, Interface, ButtonImage, and Object.

Definition at line 92 of file constructor.cpp.

```
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
    {

        loadSprites(node["textures"]);
    }

    if(node["focus"])
    {
        loadFocus(node["focus"]);
    }

    return name;
}
```

7.8.3.28 loadFocus()

```
void Container::loadFocus (
             YAML::Node node )  [protected]
```

Definition at line 218 of file constructor.cpp.

```
{
    focus[0] = node[0].as<int>();
    focus[1] = node[1].as<int>();
}
```

7.8.3.29 loadName()

```
bool Container::loadName (
             YAML::Node node )  [protected]
```

Definition at line 111 of file constructor.cpp.

```
{
    if(!node["name"])
    {
        name = "";
        return false;
    }
    name = node["name"].as<std::string>();
    return true;
}
```

7.8.3.30 loadSprites()

```
void Container::loadSprites (
             YAML::Node node )  [protected]
```

Definition at line 122 of file constructor.cpp.

```
{
    for(auto sprite : node)
    {
        if(!sprite["path"]) continue;
        if(!sprite["graphics"]) continue;

        std::string path = PASSETS::GRAPHIC_ + sprite["path"].as<std::string>();
        Image image = LoadImage(path.c_str());

        if(sprite["resize"])
        {
            int x = image.width * sprite["resize"][0].as<float>();
            int y = image.height * sprite["resize"][1].as<float>();
            ImageResize(&image, x, y);
        }

        sprites.emplace_back();
        for(auto img : sprite["graphics"])
        {
            float x, y, w, h;
            int repeat = 1;
            int gapX = 0;
            int gapY = 0;

            int dx = 1;
            int dy = 1;

            if(img["x"])
                x = img["x"].as<float>() / 100.0;
            else x = 0;
            if(img["y"])
                y = img["y"].as<float>() / 100.0;
            else y = 0;
            if(img["w"])
                w = img["w"].as<float>() / 100.0;
            else w = 1;
            if(img["h"])
                h = img["h"].as<float>() / 100.0;
            else h = 1;
            if(img["repeat"])
                repeat = img["repeat"].as<int>();
            if(img["gapX"])
                gapX = img["gapX"].as<int>();
            if(img["gapY"])
                gapY = img["gapY"].as<int>();

            if(img["dx"])
                dx = img["dx"].as<int>();
            if(dx < 0) dx = -1;
            else dx = 1;

            if(img["dy"])
                dy = img["dy"].as<int>();
            if(dy < 0) dy = -1;
            else dy = 1;

            int imgw = image.width;
            int imgh = image.height;

            if(img["axis"] && img["axis"].as<std::string>() == "horizontal")
            {
                for(float j = y; j >= 0 && j + h < 1 + 1e-2; j += dy * (gapY + h))
                {
                    for(float i = x; i >= 0 && i + w <= 1 + 1e-2 && repeat--; i += dx * (gapX + w))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }else
            {
                for(float i = x; i >= 0 && i + w <= 1 + 1e-2; i += dx * (gapX + w))
                {
                    for(float j = y; j >= 0 && j + h < 1 + 1e-2 && repeat--; j += dy * (gapY + h))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }
        }
        UnloadImage(image);
    }
}
```

7.8.3.31 moveBy() [1/2]

```
void Frame::moveBy (
            fPoint rel )   [inherited]
```

Definition at line 65 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.32 moveBy() [2/2]

```
void Frame::moveBy (
            int x,
            int y )   [inherited]
```

Definition at line 75 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.33 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
               fPoint rel )  [inherited]
```

Definition at line 43 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.34 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
               int x,
               int y )  [inherited]
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.35 moveTo() [1/2]

```
void Frame::moveTo (
               fPoint rel )  [inherited]
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.36 moveTo() [2/2]

```
void Frame::moveTo (
             int x,
             int y )  [inherited]
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.37 nextImage()

```
void Container::nextImage ( )
```

move to next state of the sprite

Definition at line 247 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]++;
    if(focus[1] >= sprites.at(focus[0]).size()) focus[1] = 0;
}
```

7.8.3.38 nextSprite()

```
void Container::nextSprite ( )
```

move to the next sprite

Definition at line 261 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]++;
    if(focus[0] >= sprites.size()) focus[0] = 0;
}
```

7.8.3.39 operator fRect()

```
Frame::operator fRect ( ) const  [inherited]
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.8.3.40 operator iRect()

```
Frame::operator iRect ( ) const  [inherited]
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.8.3.41 operator Rectangle()

```
Frame::operator Rectangle ( ) const  [inherited]
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.8.3.42 OverlappingArea() [1/2]

```
float Container::OverlappingArea (
            Container * container )
```

Definition at line 34 of file overlap.cpp.

```
{
    Rectangle rec = container->getFrame();
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.8.3.43 OverlappingArea() [2/2]

```
float Container::OverlappingArea (
            Rectangle rec )
```

Definition at line 23 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.8.3.44 plug() [1/2]

```
void Frame::plug (
            Frame * par )  [inherited]
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.8.3.45 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )  [inherited]
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.8.3.46 prevImage()

```
void Container::prevImage ( )
```

move to previous state of the sprite

Definition at line 254 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]--;
    if(focus[1] < 0) focus[1] = sprites.at(focus[0]).size() - 1;
}
```

7.8.3.47 prevSprite()

```
void Container::prevSprite ( )
```

move to the previous sprite

Definition at line 268 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]--;
    if(focus[0] < 0) focus[0] = sprites.size() - 1;
}
```

7.8.3.48 react()

```
Action * Container::react ( )  [virtual]
```

Reimplemented in ButtonImage, Game, Interface, and Object.

Definition at line 36 of file arthmetic.cpp.

```
{
    return nullptr;
}
```

7.8.3.49 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe )  [protected], [inherited]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.

```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.8.3.50 resize() [1/2]

```
void Frame::resize (
            fPoint rel )  [inherited]
```

Definition at line 85 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.51 resize() [2/2]

```
void Frame::resize (
            int w,
            int h )  [inherited]
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.52 setProbability()

```
void Container::setProbability (
            int prob )
```

Definition at line 280 of file constructor.cpp.

```
{
    probability = prob;
}
```

7.8.3.53 setRelative()

```
void Frame::setRelative (
            fRect rel )  [inherited]
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.8.3.54 show()

```
void Container::show ( )
```

Definition at line 11 of file arthmetic.cpp.

```
{
    visible = true;
}
```

7.8.3.55 toggleVisibility()

```
void Container::toggleVisibility ( )
```

Definition at line 21 of file arthmetic.cpp.

```
{
    visible = !visible;
}
```

7.8.3.56 unplug()

```
void Frame::unplug ( )  [inherited]
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.8.3.57 updateFrame()

```
void Frame::updateFrame (
            bool recursive = false )  [protected], [virtual], [inherited]
```

Reimplemented in Visual.

Definition at line 3 of file arthmetic.cpp.

```
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}
```

7.8.4 Friends And Related Symbol Documentation

7.8.4.1 changeImageAction

```
friend class changeImageAction  [friend]
```

Definition at line 22 of file container.hpp.

The documentation for this class was generated from the following files:

- src/container/include/container.hpp
- src/container/src/arthmetic.cpp
- src/container/src/constructor.cpp
- src/container/src/destructor.cpp
- src/container/src/overlap.cpp

7.9 CountDown Class Reference

count the time a playthrough takes

```
#include <countdown.hpp>
```

Public Member Functions

- CountDown (int milliseconds)
- ∼CountDown ()
- int get ()
- bool isFinished ()
- void run ()

7.9.1 Detailed Description

count the time a playthrough takes

Definition at line 12 of file countdown.hpp.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 CountDown()

```
CountDown::CountDown (
              int milliseconds )
```

Definition at line 3 of file countdown.cpp.

```
{
    start = std::chrono::system_clock::now();
    finished = false;
    elapsed_seconds = std::chrono::milliseconds(milliseconds);
}
```

7.9.2.2 ∼CountDown()

```
CountDown::∼CountDown ( )
```

Definition at line 10 of file countdown.cpp.

```
{
}
```

7.9.3 Member Function Documentation

7.9.3.1 get()

```
int CountDown::get ( )
```

Definition at line 25 of file countdown.cpp.

```
{
    return elapsed_seconds.count() * 1000;
}
```

7.9.3.2 isFinished()

```
bool CountDown::isFinished ( )
```

Definition at line 14 of file countdown.cpp.

```
{
    return finished || (std::chrono::system_clock::now() - start) > elapsed_seconds;
}
```

7.9.3.3 run()

```
void CountDown::run ( )
```

Definition at line 19 of file countdown.cpp.

```
{
    finished = false;
    start = std::chrono::system_clock::now();
}
```

The documentation for this class was generated from the following files:

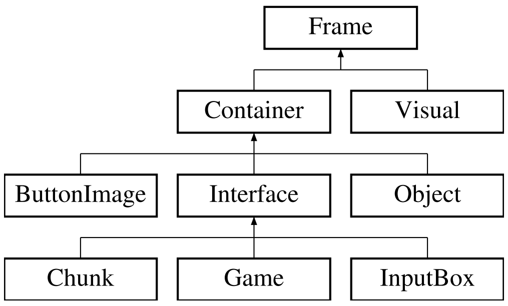
- src/utils/include/countdown.hpp
- src/utils/src/countdown.cpp

7.10 Frame Class Reference

position and size of object on screen

```
#include <frame.hpp>
```

Inheritance diagram for Frame:



Public Member Functions

- Frame (Frame ∗par, Rectangle rel)

  create a frame with a parent and a relative position
- Frame (Frame ∗self)

  clone a frame
- Frame (Rectangle rec)

  create a frame with a position and size
- ∼Frame ()

  destroy a frame
- void plug (Frame ∗par, fRect rel)

  attach a frame to a parent by relative position
- void plug (Frame ∗par)

  attach a frame to a parent by old relative position
- void unplug ()

  detach a frame from its parent
- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (fPoint rel)
- void resize (int w, int h)
- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- virtual void updateFrame (bool recursive=false)
- bool isroot () const

  return true if this frame is root
- void addSubframe (Frame ∗subframe)

  Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

  Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

7.10.1 Detailed Description

position and size of object on screen

when changing its position or size, it also changes position and size of all subframes

a subframe is relative to its parent by percentage (0.0f to 1.0f)

Definition at line 24 of file frame.hpp.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Frame() [1/3]

```
Frame::Frame (
            Frame * par,
            Rectangle rel )
```

create a frame with a parent and a relative position

Parameters

par	parent Frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 10 of file constructor.cpp.

```
{
    parent = nullptr;
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::Frame(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    parent = par;
    relative[0] = rel.x;
    relative[1] = rel.y;
    relative[2] = rel.width;
    relative[3] = rel.height;

    parent->addSubframe(this);

    updateFrame();
}
```

7.10.2.2 Frame() [2/3]

```
Frame::Frame (
            Frame * self )
```

clone a frame

Parameters

self	Frame frame to clone

Definition at line 33 of file constructor.cpp.

```
{
    parent = nullptr;
    if(self == nullptr)
    {
        throw std::runtime_error("Frame::Frame(Frame* self): self is nullptr");
        return ;
    }
    parent = self->parent;
    relative = self->relative;
    frame = self->frame;
    for(auto& i : self->subframes)
    {
        subframes.push_back(i);
    }
}
```

7.10.2.3 Frame() [3/3]

```
Frame::Frame (
            Rectangle rec )
```

create a frame with a position and size

This is a root frame

Parameters

rec	position and size in pixel

Definition at line 57 of file constructor.cpp.

```
{
    parent = nullptr;
    frame = rec;
    parent = nullptr;
    relative = {1, 1, 1, 1};
}
```

7.10.2.4 ∼Frame()

```
Frame::∼Frame ( )
```

destroy a frame

MUST NOT DELETE ANYTHING

Definition at line 10 of file destructor.cpp.

```
{
}
```

7.10.3 Member Function Documentation

7.10.3.1 addSubframe()

```
void Frame::addSubframe (
            Frame * subframe )  [protected]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.

```
{
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}
```

7.10.3.2 beginUpdate()

```
void Frame::beginUpdate ( )  [protected]
```

Definition at line 113 of file family.cpp.

```
{
    mtx.lock();
}
```

7.10.3.3 endUpdate()

```
void Frame::endUpdate ( )  [protected]
```

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.10.3.4 getCenter()

```
const fPoint & Frame::getCenter ( ) const
```

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.10.3.5 getFrame()

```
const Rectangle & Frame::getFrame ( ) const
```

Definition at line 105 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.10.3.6 getH()

```
const float & Frame::getH ( ) const
```

Definition at line 161 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}
```

7.10.3.7 getParent()

```
Frame * Frame::getParent ( )
```

Definition at line 117 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent;
}
```

7.10.3.8 getRelative()

```
const fRect & Frame::getRelative ( ) const
```

Definition at line 111 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.10.3.9 getW()

```
const float & Frame::getW ( ) const
```

Definition at line 155 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}
```

7.10.3.10 getX()

```
const float & Frame::getX ( ) const
```

Definition at line 143 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}
```

7.10.3.11 getY()

```
const float & Frame::getY ( ) const
```

Definition at line 149 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}
```

7.10.3.12 isroot()

```
bool Frame::isroot ( ) const  [protected]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.10.3.13 moveBy() [1/2]

```
void Frame::moveBy (
            fPoint rel )
```

Definition at line 65 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.14 moveBy() [2/2]

```
void Frame::moveBy (
            int x,
            int y )
```

Definition at line 75 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.15 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
            fPoint rel )
```

Definition at line 43 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.16 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
            int x,
            int y )
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.17 moveTo() [1/2]

```
void Frame::moveTo (
            fPoint rel )
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.18 moveTo() [2/2]

```
void Frame::moveTo (
            int x,
            int y )
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.19 operator fRect()

```
Frame::operator fRect ( ) const
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.10.3.20 operator iRect()

```
Frame::operator iRect ( ) const
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.10.3.21 operator Rectangle()

```
Frame::operator Rectangle ( ) const
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.10.3.22 plug() [1/2]

```
void Frame::plug (
            Frame * par )
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.10.3.23 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.
```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.10.3.24 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe )  [protected]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.
```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.10.3.25 resize() [1/2]

```
void Frame::resize (
            fPoint rel )
```

Definition at line 85 of file arthmetic.cpp.
```
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.26 resize() [2/2]

```
void Frame::resize (
               int w,
               int h )
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.27 setRelative()

```
void Frame::setRelative (
               fRect rel )
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.10.3.28 unplug()

```
void Frame::unplug ( )
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.10.3.29 updateFrame()

```
void Frame::updateFrame (
               bool recursive = false )   [protected], [virtual]
```

Reimplemented in Visual.

Definition at line 3 of file arthmetic.cpp.

```
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}
```

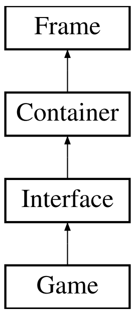
The documentation for this class was generated from the following files:

- src/frame/include/frame.hpp
- src/frame/src/arthmetic.cpp
- src/frame/src/constructor.cpp
- src/frame/src/destructor.cpp
- src/frame/src/family.cpp

7.11 Game Class Reference

```
#include <game.hpp>
```

Inheritance diagram for Game:



Public Member Functions

- Game (Frame ∗, Rectangle)
- Game (Game ∗)
- Game (Game ∗, Rectangle)
- Game (Game ∗, Frame ∗, Rectangle)
- ∼Game ()
- std::string linkContentAbsolute (std::string path) override
- Action ∗ react () override
- Action ∗ getRuntimeEvent () override
- void draw () override
- Container ∗ getContainers (int)
- int getContainersSize ()
- std::string linkContent (std::string path) override
- std::string getName ()
- void setProbability (int)
- int getProbability ()
- void chooseSprite (int)

    choose a specific sprite from a vector of sprites
- void chooseImage (int)

    choose the state of the sprite
- void chooseImage (int, int)

    choose the state of the sprite
- void nextImage ()

    move to next state of the sprite
- void prevImage ()

    move to previous state of the sprite
- void nextSprite ()

move to the next sprite
- void prevSprite ()

  move to the previous sprite
- bool isOverlapping (fPoint)
- bool isOverlapping (Rectangle)
- bool isOverlapping (Container ∗)
- float OverlappingArea (Rectangle)
- float OverlappingArea (Container ∗)
- void show ()
- void hide ()
- void toggleVisibility ()
- bool isVisible ()
- int getInstanceId ()
- void plug (Frame ∗par, fRect rel)

  attach a frame to a parent by relative position
- void plug (Frame ∗par)

  attach a frame to a parent by old relative position
- void unplug ()

  detach a frame from its parent
- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (fPoint rel)
- void resize (int w, int h)
- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- void loadChunk (YAML::Node)
- void loadCollide (YAML::Node)
- void loadEvent (YAML::Node)
- void loadAttactObject (YAML::Node)
- void loadMap ()
- void loadObject (YAML::Node)
- void loadControl (YAML::Node)
- void loadButton (YAML::Node)
- void drawNested ()
- void drawContainers ()
- bool loadName (YAML::Node node)

- void loadSprites (YAML::Node node)
- void loadFocus (YAML::Node node)
- virtual void updateFrame (bool recursive=false)
- bool isroot () const

    return true if this frame is root
- void addSubframe (Frame ∗subframe)

    Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

    Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

Friends

- class moveChunksAction

7.11.1 Detailed Description

Definition at line 16 of file game.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Game() [1/4]

```
Game::Game (
            Frame ∗ frame,
            Rectangle rect )
```

Definition at line 9 of file constructor.cpp.

```
                                        : Interface(frame, rect)
{
    initState = true;
}
```

7.11.2.2 Game() [2/4]

```
Game::Game (
            Game ∗ other )
```

Definition at line 14 of file constructor.cpp.

```
                          : Interface(other)
{
    initState = true;
}
```

7.11.2.3 Game() [3/4]

```
Game::Game (
            Game ∗ other,
            Rectangle rect )
```

Definition at line 19 of file constructor.cpp.

```
                                       : Interface(other, rect)
{
    initState = true;
}
```

7.11.2.4 Game() [4/4]

```
Game::Game (
            Game * other,
            Frame * frame,
            Rectangle rect )
```

Definition at line 24 of file constructor.cpp.

```
                                                        : Interface(other, frame, rect)
{
    initState = true;
}
```

7.11.2.5 ∼Game()

```
Game::∼Game ( )
```

Definition at line 4 of file destructor.cpp.

```
{
    for(auto &i : cache)
    {
        delete i;
    }

    for(auto &i : chunks)
    {
        delete i;
    }
}
```

7.11.3 Member Function Documentation

7.11.3.1 addSubframe()

```
void Frame::addSubframe (
            Frame * subframe )  [protected], [inherited]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.

```
{
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}
```

7.11.3.2 beginUpdate()

```
void Frame::beginUpdate ( )  [protected], [inherited]
```

Definition at line 113 of file family.cpp.

```
{
    mtx.lock();
}
```

7.11.3.3 chooseImage() [1/2]

```
void Container::chooseImage (
              int index )  [inherited]
```

choose the state of the sprite

Definition at line 231 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[1] = index;
}
```

7.11.3.4 chooseImage() [2/2]

```
void Container::chooseImage (
              int index,
              int index2 )  [inherited]
```

choose the state of the sprite

Definition at line 238 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    if(index2 < 0 || index2 >= sprites.at(index).size()) return;
    focus[0] = index;
    focus[1] = index2;
}
```

7.11.3.5 chooseSprite()

```
void Container::chooseSprite (
              int index )  [inherited]
```

choose a specific sprite from a vector of sprites

Definition at line 224 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[0] = index;
}
```

7.11.3.6 draw()

```
void Game::draw ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 3 of file arthmetic.cpp.

```
{

    drawNested();

    for(auto i = chunks.begin(); i != chunks.end(); ++i)
    {
        (*i)->draw();
    }

    drawContainers();
}
```

7.11.3.7 drawContainers()

```
void Interface::drawContainers ( )  [protected], [inherited]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    for(auto& child : containers)
    {
        child->draw();
    }
}
```

7.11.3.8 drawNested()

```
void Interface::drawNested ( )  [protected], [inherited]
```

Definition at line 3 of file arthmetic.cpp.

```
{
    for(auto& child : nested)
    {
        child->draw();
    }
}
```

7.11.3.9 endUpdate()

```
void Frame::endUpdate ( )  [protected], [inherited]
```

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.11.3.10 getCenter()

```
const fPoint & Frame::getCenter ( ) const  [inherited]
```

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.11.3.11 getContainers()

```
Container * Interface::getContainers (
              int id )  [inherited]
```

Definition at line 176 of file constructor.cpp.

```
{
    if(id < 0 || id >= containers.size()) return nullptr;
    return containers[id];
}
```

7.11.3.12 getContainersSize()

```
int Interface::getContainersSize ( ) [inherited]
```

Definition at line 182 of file constructor.cpp.

```
00183 {
00184     return containers.size();
00185 }
```

7.11.3.13 getFrame()

```
const Rectangle & Frame::getFrame ( ) const [inherited]
```

Definition at line 105 of file arthmetic.cpp.

```
00106 {
00107     std::lock_guard<std::mutex> lock(mtx);
00108     return frame;
00109 }
```

7.11.3.14 getH()

```
const float & Frame::getH ( ) const [inherited]
```

Definition at line 161 of file arthmetic.cpp.

```
00162 {
00163     std::lock_guard<std::mutex> lock(mtx);
00164     return frame.height;
00165 }
```

7.11.3.15 getInstanceId()

```
int Container::getInstanceId ( ) [inherited]
```

Definition at line 31 of file arthmetic.cpp.

```
00032 {
00033     return instance_id;
00034 }
```

7.11.3.16 getName()

```
std::string Container::getName ( ) [inherited]
```

Definition at line 275 of file constructor.cpp.

```
00276 {
00277     return name;
00278 }
```

7.11.3.17 getParent()

```
Frame * Frame::getParent ( ) [inherited]
```

Definition at line 117 of file arthmetic.cpp.

```
00118 {
00119     std::lock_guard<std::mutex> lock(mtx);
00120     return parent;
00121 }
```

7.11.3.18 getProbability()

```
int Container::getProbability ( )  [inherited]
```

Definition at line 285 of file constructor.cpp.

```
{
    return probability;
}
```

7.11.3.19 getRelative()

```
const fRect & Frame::getRelative ( ) const  [inherited]
```

Definition at line 111 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.11.3.20 getRuntimeEvent()

```
Action * Game::getRuntimeEvent ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 8 of file action.cpp.

```
{
    // if now - mapSpeedClock < 10 millisecond, return nullptr

    if(std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::system_clock::now() -
mapSpeedClock).count() < 20)
        return nullptr;
    Action* action;
    PacketAction* packet = nullptr;
    action = Interface::getRuntimeEvent();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    action = new moveChunksAction(this, mapDisplacement);
    if(packet == nullptr) packet = new PacketAction();
    packet->addAction(action);

    for(auto i : chunks)
    {
        Action* act = i->getRuntimeEvent();
        if(act == nullptr)
            continue;
        if(packet == nullptr)
            packet = new PacketAction();
        packet->addAction(act);
    }

    action = new moveObjectAction(main, mapDisplacement);
    if(packet == nullptr) packet = new PacketAction();
    packet->addAction(action);

    mapSpeedClock = std::chrono::system_clock::now();

    return packet;
}
```

7.11.3.21 getW()

```
const float & Frame::getW ( ) const  [inherited]
```

Definition at line 155 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}
```

7.11.3.22 getX()

```
const float & Frame::getX ( ) const  [inherited]
```

Definition at line 143 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}
```

7.11.3.23 getY()

```
const float & Frame::getY ( ) const  [inherited]
```

Definition at line 149 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}
```

7.11.3.24 hide()

```
void Container::hide ( )  [inherited]
```

Definition at line 16 of file arthmetic.cpp.

```
{
    visible = false;
}
```

7.11.3.25 isOverlapping() [1/3]

```
bool Container::isOverlapping (
            Container * container )  [inherited]
```

Definition at line 16 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    Rectangle rec2 = container->getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
      rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.11.3.26 isOverlapping() [2/3]

```
bool Container::isOverlapping (
            fPoint point )  [inherited]
```

Definition at line 3 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    return (point[0] >= rec.x && point[0] <= rec.x + rec.width && point[1] >= rec.y && point[1] <=
    rec.y + rec.height);

}
```

7.11.3.27 isOverlapping() [3/3]

```
bool Container::isOverlapping (
            Rectangle rec )  [inherited]
```

Definition at line 10 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
    rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.11.3.28 isroot()

```
bool Frame::isroot ( ) const  [protected], [inherited]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.11.3.29 isVisible()

```
bool Container::isVisible ( )  [inherited]
```

Definition at line 26 of file arthmetic.cpp.

```
{
    return visible;
}
```

7.11.3.30 linkContent()

```
std::string Interface::linkContent (
            std::string path )  [override], [virtual], [inherited]
```

Reimplemented from Container.

Definition at line 78 of file constructor.cpp.

```
{
    return linkContentAbsolute(PATB::INTERFACE_ + path);
}
```

7.11.3.31 linkContentAbsolute()

```
std::string Game::linkContentAbsolute (
            std::string path )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 29 of file constructor.cpp.

```
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
        loadSprites(node["textures"]);

    if(node["focus"])
        loadFocus(node["focus"]);
    else chooseImage(0, 0);

    if(node["object"])
    {
        loadObject(node["object"]);
        for(int i = 0; i < getContainersSize(); i++)
            getContainers(i)->hide();
        main = getContainers(0);
        main->show();
    }
    if(node["collide"])
        loadCollide(node["collide"]);

    if(node["chunk"])
        loadChunk(node["chunk"]);

    if(node["attach-object"])
        loadAttactObject(node["attach-object"]);

    if(node["control"])
        loadControl(node["control"]);

    if(node["event"])
        loadEvent(node["event"]);

    if(node["button"])
        loadButton(node["button"]);
    return getName();
}
```

7.11.3.32 loadAttactObject()

```
void Game::loadAttactObject (
            YAML::Node node )  [protected]
```

Definition at line 143 of file constructor.cpp.

```
{
    for(auto i : node)
    {
        int id = i["chunk"].as<int>();
        int objID = i["object"][0].as<int>();
        int prob = i["object"][1].as<int>();
        Container* container = getContainers(objID);
        container->setProbability(prob);
        cache[id]->addVisiter(container);
    }
}
```

7.11.3.33 loadButton()

```
void Interface::loadButton (
            YAML::Node node )  [protected], [inherited]
```

Definition at line 159 of file constructor.cpp.

```
00160 {
00161     for(auto i : node)
00162     {
00163         Rectangle rel({0, 0, 0, 0});
00164         if(i["x"]) rel.x = i["x"].as<float>() / 100;
00165         if(i["y"]) rel.y = i["y"].as<float>() / 100;
00166         if(i["w"]) rel.width = i["w"].as<float>() / 100;
00167         if(i["h"]) rel.height = i["h"].as<float>() / 100;
00168         ButtonImage *obj;
00169         obj = new ButtonImage(this, rel);
00170         obj->linkContent(i["path"].as<std::string>());
00171         obj->show();
00172         containers.push_back(obj);
00173     }
00174 }
```

7.11.3.34 loadChunk()

```
void Game::loadChunk (
            YAML::Node node )  [protected]
```

Definition at line 113 of file constructor.cpp.

```
00114 {
00115     for(auto i : node)
00116     {
00117         float x = 0, y = 0, w = 1, h = 1;
00118         int repeat = 1;
00119         std::string path = i["file"].as<std::string>();
00120         if(i["x"]) x = i["x"].as<float>() / 100;
00121         if(i["y"]) y = i["y"].as<float>() / 100;
00122         if(i["w"]) w = i["w"].as<float>() / 100;
00123         if(i["h"]) h = i["h"].as<float>() / 100;
00124         if(i["repeat"]) repeat = i["repeat"].as<int>();
00125         fPoint direction = {1, 0};
00126         float velo = 0.002;
00127         if(i["velocity"])
00128         {
00129             velo = i["velocity"][0].as<float>();
00130             direction = {i["velocity"][1].as<float>(), i["velocity"][2].as<float>()};
00131         }
00132         float angle = VECTOR2D::getAngle(direction);
00133         fPoint displacement = {velo * cos(angle), velo * sin(angle)};
00134         Chunk* chunk = new Chunk(this, {x, y, w, h});
00135         chunk->linkContent(path);
00136         chunk->setVelocity(displacement);
00137         cache.push_back(chunk);
00138         while(--repeat > 0)
00139             cache.push_back(new Chunk(cache[0]));
00140     }
00141 }
```

7.11.3.35 loadCollide()

```
void Game::loadCollide (
            YAML::Node node )  [protected]
```

Definition at line 69 of file constructor.cpp.

```
00070 {
00071 }
```

7.11.3.36 loadControl()

```
void Interface::loadControl (
            YAML::Node node )  [protected], [inherited]
```

Definition at line 134 of file constructor.cpp.

```
00135 {
00136     for(auto stroke : node)
00137     {
```

```
00138         KeyStroke* k = new KeyStroke();
00139         for(auto key : stroke["key"])
00140         {
00141             k->add(toKey(key.as<std::string>()));
00142         }
00143         std::string action = stroke["action"].as<std::string>();
00144 
00145         if(action == "move-object")
00146         {
00147             int id = stroke["args"][0].as<int>();
00148             float v = stroke["args"][1].as<float>() / 100.0;
00149             float x = stroke["args"][2].as<float>();
00150             float y = stroke["args"][3].as<float>();
00151             moveObjectAction* action = new moveObjectAction(containers[id], fPoint({x, y}), v);
00152             k->addAction(action);
00153         }
00154 
00155         keystrokes.push_back(k);
00156     }
00157 }
```

7.11.3.37 loadEvent()

```
void Game::loadEvent (
              YAML::Node node )  [protected]
```

Definition at line 156 of file constructor.cpp.

```
00157 {
00158     if(node["map-speed"])
00159     {
00160         mapSpeed = node["map-speed"].as<float>();
00161     }
00162     if(node["map-direction"])
00163     {
00164         mapDirection[0] = node["map-direction"][0].as<float>();
00165         mapDirection[1] = node["map-direction"][1].as<float>();
00166     }
00167     float angle = VECTOR2D::getAngle(mapDirection);
00168     std::cout « "hehe: " « angle « std::endl;
00169     mapDisplacement[0] = mapSpeed * cos(angle);
00170     mapDisplacement[1] = mapSpeed * sin(angle);
00171 }
```

7.11.3.38 loadFocus()

```
void Container::loadFocus (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 218 of file constructor.cpp.

```
00219 {
00220     focus[0] = node[0].as<int>();
00221     focus[1] = node[1].as<int>();
00222 }
```

7.11.3.39 loadMap()

```
void Game::loadMap ( )  [protected]
```

Definition at line 73 of file constructor.cpp.

```
00074 {
00075     if(cache.empty()) return ;
00076     while(!chunks.empty())
00077     {
00078         fRect rec = chunks.back()->getRelative();
00079         if(rec[1] > 1) chunks.pop_back();
00080         else break;
00081     }
00082     if(chunks.empty())
00083     {
        Rectangle rel;
        rel.width = cache[0]->getRelative()[2];
        rel.height = cache[0]->getRelative()[3];
        rel.x = 0;
        rel.y = (1.01 - rel.height);

        Chunk* chunk = new Chunk(cache[0], this, rel);
        chunks.push_front(chunk);
        for(int i = 0; i < 3; i++)
        {
            rel.y += 0.005 - rel.height;
            chunk = new Chunk(cache[0], this, rel);
            chunks.push_front(chunk);
        }
    }
    while(chunks.front()->getRelative()[1] > 0)
    {
        Rectangle rel;
        rel.width = chunks.front()->getRelative()[2];
        rel.height = chunks.front()->getRelative()[3];
        rel.x = 0;
        rel.y = (chunks.front()->getRelative()[1] + 0.005 - rel.height);

        int id = GetRandomValue(0, cache.size() - 1);
        Chunk* chunk = new Chunk(cache[id], this, rel);
        chunks.push_front(chunk);
    }
}
```

7.11.3.40 loadName()

```
bool Container::loadName (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 111 of file constructor.cpp.

```
{
    if(!node["name"])
    {
        name = "";
        return false;
    }
    name = node["name"].as<std::string>();
    return true;
}
```

7.11.3.41 loadObject()

```
void Interface::loadObject (
              YAML::Node node )  [protected], [inherited]
```

Definition at line 117 of file constructor.cpp.

```
{
    for(auto i : node)
    {
        Rectangle rel({0, 0, 0, 0});
        if(i["x"]) rel.x = i["x"].as<float>() / 100;
        if(i["y"]) rel.y = i["y"].as<float>() / 100;
        if(i["w"]) rel.width = i["w"].as<float>() / 100;
        if(i["h"]) rel.height = i["h"].as<float>() / 100;
        Container *obj;
        obj = new Object(this, rel);
        obj->linkContent(i["path"].as<std::string>());
        containers.push_back(obj);
    }
}
```

7.11.3.42 loadSprites()

```
void Container::loadSprites (
            YAML::Node node )  [protected], [inherited]
```

Definition at line 122 of file constructor.cpp.

```
{
    for(auto sprite : node)
    {
        if(!sprite["path"]) continue;
        if(!sprite["graphics"]) continue;

        std::string path = PASSETS::GRAPHIC_ + sprite["path"].as<std::string>();
        Image image = LoadImage(path.c_str());

        if(sprite["resize"])
        {
            int x = image.width * sprite["resize"][0].as<float>();
            int y = image.height * sprite["resize"][1].as<float>();
            ImageResize(&image, x, y);
        }

        sprites.emplace_back();
        for(auto img : sprite["graphics"])
        {
            float x, y, w, h;
            int repeat = 1;
            int gapX = 0;
            int gapY = 0;

            int dx = 1;
            int dy = 1;

            if(img["x"])
                x = img["x"].as<float>() / 100.0;
            else x = 0;
            if(img["y"])
                y = img["y"].as<float>() / 100.0;
            else y = 0;
            if(img["w"])
                w = img["w"].as<float>() / 100.0;
            else w = 1;
            if(img["h"])
                h = img["h"].as<float>() / 100.0;
            else h = 1;
            if(img["repeat"])
                repeat = img["repeat"].as<int>();
            if(img["gapX"])
                gapX = img["gapX"].as<int>();
            if(img["gapY"])
                gapY = img["gapY"].as<int>();

            if(img["dx"])
                dx = img["dx"].as<int>();
            if(dx < 0) dx = -1;
            else dx = 1;

            if(img["dy"])
                dy = img["dy"].as<int>();
            if(dy < 0) dy = -1;
            else dy = 1;

            int imgw = image.width;
            int imgh = image.height;

            if(img["axis"] && img["axis"].as<std::string>() == "horizontal")
            {
                for(float j = y; j >= 0 && j + h < 1 + 1e-2; j += dy * (gapY + h))
                {
                    for(float i = x; i >= 0 && i + w <= 1 + 1e-2 && repeat--; i += dx * (gapX + w))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }else
            {
                for(float i = x; i >= 0 && i + w <= 1 + 1e-2; i += dx * (gapX + w))
                {
```

```
00201                     for(float j = y; j >= 0 && j + h < 1 + 1e-2 && repeat--; j += dy * (gapY + h))
00202                     {
00203                         Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
00204                         Image img2 = ImageFromImage(image, rect);
00205                         Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
00206                         Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
00207                         sprites.back().push_back(vis);
00208 
00209                         UnloadImage(img2);
00210                     }
00211                 }
00212             }
00213         }
00214         UnloadImage(image);
00215     }
00216 }
```

7.11.3.43 moveBy() [1/2]

```
void Frame::moveBy (
              fPoint rel )  [inherited]
```

Definition at line 65 of file arthmetic.cpp.

```
00066 {
00067     if(isroot()) return ;
00068     mtx.lock();
00069     relative[0] += rel[0];
00070     relative[1] += rel[1];
00071     mtx.unlock();
00072     updateFrame(true);
00073 }
```

7.11.3.44 moveBy() [2/2]

```
void Frame::moveBy (
              int x,
              int y )  [inherited]
```

Definition at line 75 of file arthmetic.cpp.

```
00076 {
00077     if(parent != nullptr) return ;
00078     mtx.lock();
00079     frame.x += x;
00080     frame.y += y;
00081     mtx.unlock();
00082     updateFrame(true);
00083 }
```

7.11.3.45 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
              fPoint rel )  [inherited]
```

Definition at line 43 of file arthmetic.cpp.

```
00044 {
00045     if(isroot()) return ;
00046     mtx.lock();
00047     fPoint center = getCenter();
00048     relative[0] += rel[0] - center[0];
00049     relative[1] += rel[1] - center[1];
00050     mtx.unlock();
00051     updateFrame(true);
00052 }
```

7.11.3.46 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
             int x,
             int y ) [inherited]
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.11.3.47 moveTo() [1/2]

```
void Frame::moveTo (
             fPoint rel ) [inherited]
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.11.3.48 moveTo() [2/2]

```
void Frame::moveTo (
             int x,
             int y ) [inherited]
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.11.3.49 nextImage()

```
void Container::nextImage ( ) [inherited]
```

move to next state of the sprite

Definition at line 247 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]++;
    if(focus[1] >= sprites.at(focus[0]).size()) focus[1] = 0;
}
```

7.11.3.50 nextSprite()

```
void Container::nextSprite ( )  [inherited]
```

move to the next sprite

Definition at line 261 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]++;
    if(focus[0] >= sprites.size()) focus[0] = 0;
}
```

7.11.3.51 operator fRect()

```
Frame::operator fRect ( ) const  [inherited]
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.11.3.52 operator iRect()

```
Frame::operator iRect ( ) const  [inherited]
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.11.3.53 operator Rectangle()

```
Frame::operator Rectangle ( ) const  [inherited]
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.11.3.54 OverlappingArea() [1/2]

```
float Container::OverlappingArea (
             Container * container )  [inherited]
```

Definition at line 34 of file overlap.cpp.

```
{
    Rectangle rec = container->getFrame();
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.11.3.55 OverlappingArea() [2/2]

```
float Container::OverlappingArea (
            Rectangle rec )  [inherited]
```

Definition at line 23 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.11.3.56 plug() [1/2]

```
void Frame::plug (
            Frame * par )  [inherited]
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.11.3.57 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )  [inherited]
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.11.3.58 prevImage()

```
void Container::prevImage ( ) [inherited]
```

move to previous state of the sprite

Definition at line 254 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]--;
    if(focus[1] < 0) focus[1] = sprites.at(focus[0]).size() - 1;
}
```

7.11.3.59 prevSprite()

```
void Container::prevSprite ( ) [inherited]
```

move to the previous sprite

Definition at line 268 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]--;
    if(focus[0] < 0) focus[0] = sprites.size() - 1;
}
```

7.11.3.60 react()

```
Action * Game::react ( ) [override], [virtual]
```

Reimplemented from Container.

Definition at line 3 of file action.cpp.

```
{
    return Interface::react();
}
```

7.11.3.61 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe ) [protected], [inherited]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.

```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.11.3.62 resize() [1/2]

```
void Frame::resize (
              fPoint rel )   [inherited]
```

Definition at line 85 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.11.3.63 resize() [2/2]

```
void Frame::resize (
              int w,
              int h )   [inherited]
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.11.3.64 setProbability()

```
void Container::setProbability (
              int prob )   [inherited]
```

Definition at line 280 of file constructor.cpp.

```
{
    probability = prob;
}
```

7.11.3.65 setRelative()

```
void Frame::setRelative (
              fRect rel )  [inherited]
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.11.3.66 show()

```
void Container::show ( )  [inherited]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    visible = true;
}
```

7.11.3.67 toggleVisibility()

```
void Container::toggleVisibility ( )  [inherited]
```

Definition at line 21 of file arthmetic.cpp.

```
{
    visible = !visible;
}
```

7.11.3.68 unplug()

```
void Frame::unplug ( )  [inherited]
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.11.3.69 updateFrame()

```
void Frame::updateFrame (
              bool recursive = false )  [protected], [virtual], [inherited]
```

Reimplemented in Visual.

Definition at line 3 of file arthmetic.cpp.

```
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}
```

7.11.4 Friends And Related Symbol Documentation

7.11.4.1 moveChunksAction

```
friend class moveChunksAction  [friend]
```

Definition at line 19 of file game.hpp.

The documentation for this class was generated from the following files:

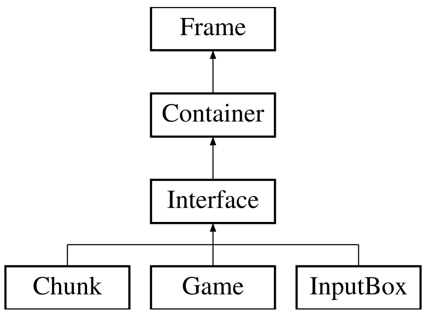
- src/game/include/game.hpp
- src/game/src/action.cpp
- src/game/src/arthmetic.cpp
- src/game/src/constructor.cpp
- src/game/src/destructor.cpp

7.12 Interface Class Reference

where user can interact with the game

```
#include <interface.hpp>
```

Inheritance diagram for Interface:



Public Member Functions

- Interface (Frame ∗, Rectangle)
- Interface (Interface ∗)
- Interface (Interface ∗, Rectangle)
- Interface (Interface ∗, Frame ∗, Rectangle)
- ∼Interface ()
- Container ∗ getContainers (int)
- int getContainersSize ()
- std::string linkContent (std::string path) override
- std::string linkContentAbsolute (std::string path) override
- Action ∗ react () override
- Action ∗ getRuntimeEvent () override
- void draw () override
- std::string getName ()
- void setProbability (int)

- int getProbability ()
- void chooseSprite (int)

    choose a specific sprite from a vector of sprites
- void chooseImage (int)

    choose the state of the sprite
- void chooseImage (int, int)

    choose the state of the sprite
- void nextImage ()

    move to next state of the sprite
- void prevImage ()

    move to previous state of the sprite
- void nextSprite ()

    move to the next sprite
- void prevSprite ()

    move to the previous sprite
- bool isOverlapping (fPoint)
- bool isOverlapping (Rectangle)
- bool isOverlapping (Container ∗)
- float OverlappingArea (Rectangle)
- float OverlappingArea (Container ∗)
- void show ()
- void hide ()
- void toggleVisibility ()
- bool isVisible ()
- int getInstanceId ()
- void plug (Frame ∗par, fRect rel)

    attach a frame to a parent by relative position
- void plug (Frame ∗par)

    attach a frame to a parent by old relative position
- void unplug ()

    detach a frame from its parent
- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (fPoint rel)
- void resize (int w, int h)
- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- void loadObject (YAML::Node)
- void loadControl (YAML::Node)
- void loadButton (YAML::Node)
- void drawNested ()
- void drawContainers ()
- bool loadName (YAML::Node node)
- void loadSprites (YAML::Node node)
- void loadFocus (YAML::Node node)
- virtual void updateFrame (bool recursive=false)
- bool isroot () const

  return true if this frame is root
- void addSubframe (Frame ∗subframe)

  Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

  Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

Friends

- class moveObjectAction

7.12.1 Detailed Description

where user can interact with the game

manages containers, all actions, subframes etc.

Definition at line 20 of file interface.hpp.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Interface() [1/4]

```
Interface::Interface (
            Frame ∗ frame,
            Rectangle rect )
```

Definition at line 8 of file constructor.cpp.

```
00008                                        : Container(frame, rect)
00009 {
00010 }
```

7.12.2.2 Interface() [2/4]

```
Interface::Interface (
            Interface * other )
```

Definition at line 12 of file constructor.cpp.

```
                                   : Container(other)
{
    for(auto i : other->nested)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        nested.push_back(new Interface(i, this, rel));
    }
    for(auto i : other->containers)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        containers.push_back(new Container(i, this, rel));
    }
}
```

7.12.2.3 Interface() [3/4]

```
Interface::Interface (
            Interface * other,
            Rectangle rect )
```

Definition at line 34 of file constructor.cpp.

```
                                                   : Container(other, rect)
{
    for(auto i : other->nested)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        nested.push_back(new Interface(i, this, rel));
    }
    for(auto i : other->containers)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        containers.push_back(new Container(i, this, rel));
    }
}
```

7.12.2.4 Interface() [4/4]

```
Interface::Interface (
            Interface * other,
            Frame * frame,
            Rectangle rect )
```

Definition at line 56 of file constructor.cpp.

```
                                                           : Container(other, frame, rect)
{
    for(auto i : other->nested)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        nested.push_back(new Interface(i, this, rel));
    }
    for(auto i : other->containers)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        containers.push_back(new Container(i, this, rel));
    }
}
```

7.12.2.5 ∼Interface()

```
Interface::∼Interface ( )
```

Definition at line 4 of file destructor.cpp.

```
{
    for (auto& i : containers)
        delete i;
    containers.clear();

    for (auto& i : nested)
        delete i;
    nested.clear();

    for (auto& i : keystrokes)
        delete i;
}
```

7.12.3 Member Function Documentation

7.12.3.1 addSubframe()

```
void Frame::addSubframe (
            Frame * subframe )  [protected], [inherited]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.

```
{
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}
```

7.12.3.2 beginUpdate()

```
void Frame::beginUpdate ( )  [protected], [inherited]
```

Definition at line 113 of file family.cpp.

```
{
    mtx.lock();
}
```

7.12.3.3 chooseImage() [1/2]

```
void Container::chooseImage (
              int index )  [inherited]
```

choose the state of the sprite

Definition at line 231 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[1] = index;
}
```

7.12.3.4 chooseImage() [2/2]

```
void Container::chooseImage (
              int index,
              int index2 )  [inherited]
```

choose the state of the sprite

Definition at line 238 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    if(index2 < 0 || index2 >= sprites.at(index).size()) return;
    focus[0] = index;
    focus[1] = index2;
}
```

7.12.3.5 chooseSprite()

```
void Container::chooseSprite (
              int index )  [inherited]
```

choose a specific sprite from a vector of sprites

Definition at line 224 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[0] = index;
}
```

7.12.3.6 draw()

```
void Interface::draw ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 19 of file arthmetic.cpp.

```
{
    Container::draw();

    drawNested();

    drawContainers();


}
```

7.12.3.7 drawContainers()

```
void Interface::drawContainers ( )  [protected]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    for(auto& child : containers)
    {
        child->draw();
    }
}
```

7.12.3.8 drawNested()

```
void Interface::drawNested ( )  [protected]
```

Definition at line 3 of file arthmetic.cpp.

```
{
    for(auto& child : nested)
    {
        child->draw();
    }
}
```

7.12.3.9 endUpdate()

```
void Frame::endUpdate ( )  [protected], [inherited]
```

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.12.3.10 getCenter()

```
const fPoint & Frame::getCenter ( ) const  [inherited]
```

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.12.3.11 getContainers()

```
Container * Interface::getContainers (
            int id )
```

Definition at line 176 of file constructor.cpp.

```
{
    if(id < 0 || id >= containers.size()) return nullptr;
    return containers[id];
}
```

7.12.3.12 getContainersSize()

```
int Interface::getContainersSize ( )
```

Definition at line 182 of file constructor.cpp.

```
{
    return containers.size();
}
```

7.12.3.13 getFrame()

```
const Rectangle & Frame::getFrame ( ) const  [inherited]
```

Definition at line 105 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.12.3.14 getH()

```
const float & Frame::getH ( ) const  [inherited]
```

Definition at line 161 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}
```

7.12.3.15 getInstanceId()

```
int Container::getInstanceId ( )  [inherited]
```

Definition at line 31 of file arthmetic.cpp.

```
{
    return instance_id;
}
```

7.12.3.16 getName()

```
std::string Container::getName ( )  [inherited]
```

Definition at line 275 of file constructor.cpp.

```
{
    return name;
}
```

7.12.3.17 getParent()

```
Frame * Frame::getParent ( )  [inherited]
```

Definition at line 117 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent;
}
```

7.12.3.18 getProbability()

```
int Container::getProbability ( )  [inherited]
```

Definition at line 285 of file constructor.cpp.

```
{
    return probability;
}
```

7.12.3.19 getRelative()

```
const fRect & Frame::getRelative ( ) const  [inherited]
```

Definition at line 111 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.12.3.20 getRuntimeEvent()

```
Action * Interface::getRuntimeEvent ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 4 of file action.cpp.

```
{
    PacketAction* packet = nullptr;
    Action* action = Container::getRuntimeEvent();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    for(auto i : nested)
    {
        action = i->getRuntimeEvent();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    for(auto i : containers)
    {
        action = i->getRuntimeEvent();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    return packet;
}
```

7.12.3.21 getW()

```
const float & Frame::getW ( ) const  [inherited]
```

Definition at line 155 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}
```

7.12.3.22 getX()

```
const float & Frame::getX ( ) const  [inherited]
```

Definition at line 143 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}
```

7.12.3.23 getY()

```
const float & Frame::getY ( ) const  [inherited]
```

Definition at line 149 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}
```

7.12.3.24 hide()

```
void Container::hide ( )  [inherited]
```

Definition at line 16 of file arthmetic.cpp.

```
{
    visible = false;
}
```

7.12.3.25 isOverlapping() [1/3]

```
bool Container::isOverlapping (
            Container * container )  [inherited]
```

Definition at line 16 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    Rectangle rec2 = container->getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
      rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.12.3.26 isOverlapping() [2/3]

```
bool Container::isOverlapping (
            fPoint point )  [inherited]
```

Definition at line 3 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    return (point[0] >= rec.x && point[0] <= rec.x + rec.width && point[1] >= rec.y && point[1] <=
      rec.y + rec.height);

}
```

7.12.3.27 isOverlapping() [3/3]

```
bool Container::isOverlapping (
            Rectangle rec )  [inherited]
```

Definition at line 10 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
  rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.12.3.28 isroot()

```
bool Frame::isroot ( ) const  [protected], [inherited]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.12.3.29 isVisible()

```
bool Container::isVisible ( )  [inherited]
```

Definition at line 26 of file arthmetic.cpp.

```
{
    return visible;
}
```

7.12.3.30 linkContent()

```
std::string Interface::linkContent (
            std::string path )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 78 of file constructor.cpp.

```
{
    return linkContentAbsolute(PATB::INTERFACE_ + path);
}
```

7.12.3.31 linkContentAbsolute()

```
std::string Interface::linkContentAbsolute (
              std::string path )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 83 of file constructor.cpp.

```
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
        loadSprites(node["textures"]);

    if(node["focus"])
        loadFocus(node["focus"]);
    else chooseImage(0, 0);

    if(node["object"])
        loadObject(node["object"]);

    if(node["control"])
        loadControl(node["control"]);

    if(node["button"])
        loadButton(node["button"]);

//    if(node["collide"])
//        loadCollide(node["collide"]);

//    if(node["chunk"])
//        loadChunk(node["chunk"]);


//    if(node["event"])
//        loadEvent(node["event"]);

    return getName();
}
```

7.12.3.32 loadButton()

```
void Interface::loadButton (
              YAML::Node node )  [protected]
```

Definition at line 159 of file constructor.cpp.

```
{
    for(auto i : node)
    {
        Rectangle rel({0, 0, 0, 0});
        if(i["x"]) rel.x = i["x"].as<float>() / 100;
        if(i["y"]) rel.y = i["y"].as<float>() / 100;
        if(i["w"]) rel.width = i["w"].as<float>() / 100;
        if(i["h"]) rel.height = i["h"].as<float>() / 100;
        ButtonImage *obj;
        obj = new ButtonImage(this, rel);
        obj->linkContent(i["path"].as<std::string>());
        obj->show();
        containers.push_back(obj);
    }
}
```

7.12.3.33 loadControl()

```
void Interface::loadControl (
              YAML::Node node )  [protected]
```

Definition at line 134 of file constructor.cpp.

```
{
    for(auto stroke : node)
    {
        KeyStroke* k = new KeyStroke();
        for(auto key : stroke["key"])
        {
            k->add(toKey(key.as<std::string>()));
        }
        std::string action = stroke["action"].as<std::string>();

        if(action == "move-object")
        {
            int id = stroke["args"][0].as<int>();
            float v = stroke["args"][1].as<float>() / 100.0;
            float x = stroke["args"][2].as<float>();
            float y = stroke["args"][3].as<float>();
            moveObjectAction* action = new moveObjectAction(containers[id], fPoint({x, y}), v);
            k->addAction(action);
        }

        keystrokes.push_back(k);
    }
}
```

7.12.3.34 loadFocus()

```
void Container::loadFocus (
            YAML::Node node )  [protected], [inherited]
```

Definition at line 218 of file constructor.cpp.

```
{
    focus[0] = node[0].as<int>();
    focus[1] = node[1].as<int>();
}
```

7.12.3.35 loadName()

```
bool Container::loadName (
            YAML::Node node )  [protected], [inherited]
```

Definition at line 111 of file constructor.cpp.

```
{
    if(!node["name"])
    {
        name = "";
        return false;
    }
    name = node["name"].as<std::string>();
    return true;
}
```

7.12.3.36 loadObject()

```
void Interface::loadObject (
            YAML::Node node )  [protected]
```

Definition at line 117 of file constructor.cpp.

```
{
    for(auto i : node)
    {
        Rectangle rel({0, 0, 0, 0});
        if(i["x"]) rel.x = i["x"].as<float>() / 100;
        if(i["y"]) rel.y = i["y"].as<float>() / 100;
        if(i["w"]) rel.width = i["w"].as<float>() / 100;
        if(i["h"]) rel.height = i["h"].as<float>() / 100;
        Container *obj;
        obj = new Object(this, rel);
        obj->linkContent(i["path"].as<std::string>());
        containers.push_back(obj);
    }
}
```

7.12.3.37 loadSprites()

```
void Container::loadSprites (
            YAML::Node node ) [protected], [inherited]
```

Definition at line 122 of file constructor.cpp.

```
{
    for(auto sprite : node)
    {
        if(!sprite["path"]) continue;
        if(!sprite["graphics"]) continue;

        std::string path = PASSETS::GRAPHIC_ + sprite["path"].as<std::string>();
        Image image = LoadImage(path.c_str());

        if(sprite["resize"])
        {
            int x = image.width * sprite["resize"][0].as<float>();
            int y = image.height * sprite["resize"][1].as<float>();
            ImageResize(&image, x, y);
        }

        sprites.emplace_back();
        for(auto img : sprite["graphics"])
        {
            float x, y, w, h;
            int repeat = 1;
            int gapX = 0;
            int gapY = 0;

            int dx = 1;
            int dy = 1;

            if(img["x"])
                x = img["x"].as<float>() / 100.0;
            else x = 0;
            if(img["y"])
                y = img["y"].as<float>() / 100.0;
            else y = 0;
            if(img["w"])
                w = img["w"].as<float>() / 100.0;
            else w = 1;
            if(img["h"])
                h = img["h"].as<float>() / 100.0;
            else h = 1;
            if(img["repeat"])
                repeat = img["repeat"].as<int>();
            if(img["gapX"])
                gapX = img["gapX"].as<int>();
            if(img["gapY"])
                gapY = img["gapY"].as<int>();

            if(img["dx"])
                dx = img["dx"].as<int>();
            if(dx < 0) dx = -1;
            else dx = 1;

            if(img["dy"])
                dy = img["dy"].as<int>();
            if(dy < 0) dy = -1;
            else dy = 1;

            int imgw = image.width;
            int imgh = image.height;

            if(img["axis"] && img["axis"].as<std::string>() == "horizontal")
            {
                for(float j = y; j >= 0 && j + h < 1 + 1e-2; j += dy * (gapY + h))
                {
                    for(float i = x; i >= 0 && i + w <= 1 + 1e-2 && repeat--; i += dx * (gapX + w))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }else
            {
                for(float i = x; i >= 0 && i + w <= 1 + 1e-2; i += dx * (gapX + w))
                {
                    for(float j = y; j >= 0 && j + h < 1 + 1e-2 && repeat--; j += dy * (gapY + h))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }
        }
        UnloadImage(image);
    }
}
```

7.12.3.38 moveBy() [1/2]

```
void Frame::moveBy (
            fPoint rel ) [inherited]
```

Definition at line 65 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.39 moveBy() [2/2]

```
void Frame::moveBy (
            int x,
            int y ) [inherited]
```

Definition at line 75 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.40 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
            fPoint rel ) [inherited]
```

Definition at line 43 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.41 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
            int x,
            int y )  [inherited]
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.42 moveTo() [1/2]

```
void Frame::moveTo (
            fPoint rel )  [inherited]
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.43 moveTo() [2/2]

```
void Frame::moveTo (
            int x,
            int y )  [inherited]
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.44 nextImage()

```
void Container::nextImage ( )  [inherited]
```

move to next state of the sprite

Definition at line 247 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]++;
    if(focus[1] >= sprites.at(focus[0]).size()) focus[1] = 0;
}
```

7.12.3.45 nextSprite()

```
void Container::nextSprite ( )  [inherited]
```

move to the next sprite

Definition at line 261 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]++;
    if(focus[0] >= sprites.size()) focus[0] = 0;
}
```

7.12.3.46 operator fRect()

```
Frame::operator fRect ( ) const  [inherited]
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.12.3.47 operator iRect()

```
Frame::operator iRect ( ) const  [inherited]
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.12.3.48 operator Rectangle()

```
Frame::operator Rectangle ( ) const  [inherited]
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.12.3.49 OverlappingArea() [1/2]

```
float Container::OverlappingArea (
            Container * container )  [inherited]
```

Definition at line 34 of file overlap.cpp.

```
{
    Rectangle rec = container->getFrame();
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.12.3.50 OverlappingArea() [2/2]

```
float Container::OverlappingArea (
            Rectangle rec )  [inherited]
```

Definition at line 23 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.12.3.51 plug() [1/2]

```
void Frame::plug (
            Frame * par )  [inherited]
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.12.3.52 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )  [inherited]
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.12.3.53 prevImage()

`void Container::prevImage ( )  [inherited]`

move to previous state of the sprite

Definition at line 254 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]--;
    if(focus[1] < 0) focus[1] = sprites.at(focus[0]).size() - 1;
}
```

7.12.3.54 prevSprite()

`void Container::prevSprite ( )  [inherited]`

move to the previous sprite

Definition at line 268 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]--;
    if(focus[0] < 0) focus[0] = sprites.size() - 1;
}
```

7.12.3.55 react()

`Action * Interface::react ( )  [override], [virtual]`

Reimplemented from Container.

Definition at line 38 of file action.cpp.

```
{
    if(!isVisible()) return nullptr;
    PacketAction* packet = nullptr;

    Action* action = Container::react();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    for(auto i : keystrokes)
    {
        Action* action = i->react();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    for(auto i : containers)
    {
        Action* action = i->react();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    return packet;
}
```

7.12.3.56 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe )  [protected], [inherited]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.

```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.12.3.57 resize() [1/2]

```
void Frame::resize (
            fPoint rel )  [inherited]
```

Definition at line 85 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.58 resize() [2/2]

```
void Frame::resize (
            int w,
            int h )  [inherited]
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.59 setProbability()

```
void Container::setProbability (
             int prob )  [inherited]
```

Definition at line 280 of file constructor.cpp.

```
{
    probability = prob;
}
```

7.12.3.60 setRelative()

```
void Frame::setRelative (
             fRect rel )  [inherited]
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.12.3.61 show()

```
void Container::show ( )  [inherited]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    visible = true;
}
```

7.12.3.62 toggleVisibility()

```
void Container::toggleVisibility ( )  [inherited]
```

Definition at line 21 of file arthmetic.cpp.

```
{
    visible = !visible;
}
```

7.12.3.63 unplug()

```
void Frame::unplug ( )  [inherited]
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.12.3.64 updateFrame()

```
void Frame::updateFrame (
            bool recursive = false )  [protected], [virtual], [inherited]
```

Reimplemented in Visual.

Definition at line 3 of file arthmetic.cpp.

```
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}
```

7.12.4 Friends And Related Symbol Documentation

7.12.4.1 moveObjectAction

```
friend class moveObjectAction  [friend]
```

Definition at line 23 of file interface.hpp.

The documentation for this class was generated from the following files:

- src/interface/include/interface.hpp
- src/interface/src/action.cpp
- src/interface/src/arthmetic.cpp
- src/interface/src/constructor.cpp
- src/interface/src/destructor.cpp

7.13 KeyStroke Class Reference

manages the link between a key and the actions it performs

```
#include <keystroke.hpp>
```

Public Member Functions

- KeyStroke ()
- KeyStroke (std::vector< int >)
- ∼KeyStroke ()
- int size ()
- void add (unsigned char)
- void setAction (std::vector< Action ∗ >)
- void addAction (Action ∗)
- void chooseAction (int)
- int getCurrent (int)
- void nextAction ()
- Action ∗ react ()

7.13.1 Detailed Description

manages the link between a key and the actions it performs

Definition at line 16 of file keystroke.hpp.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 KeyStroke() [1/2]

```
KeyStroke::KeyStroke ( )
```

Definition at line 6 of file keystroke.cpp.

```
{
    id = 0;
}
```

7.13.2.2 KeyStroke() [2/2]

```
KeyStroke::KeyStroke (
              std::vector< int > k )
```

Definition at line 11 of file keystroke.cpp.

```
{
    key = k;
    id = 0;
}
```

7.13.2.3 ∼KeyStroke()

```
KeyStroke::∼KeyStroke ( )
```

Definition at line 17 of file keystroke.cpp.

```
{
    for(auto &a : action)
    {
        delete a;
    }
}
```

7.13.3 Member Function Documentation

7.13.3.1 add()

```
void KeyStroke::add (
              unsigned char k )
```

Definition at line 30 of file keystroke.cpp.

```
{
    key.push_back(k);
}
```

7.13.3.2 addAction()

```
void KeyStroke::addAction (
            Action * a )
```

Definition at line 39 of file keystroke.cpp.

```
{
    action.push_back(a);
}
```

7.13.3.3 chooseAction()

```
void KeyStroke::chooseAction (
            int i )
```

Definition at line 52 of file keystroke.cpp.

```
{
    id = i;
}
```

7.13.3.4 getCurrent()

```
int KeyStroke::getCurrent (
            int i )
```

Definition at line 57 of file keystroke.cpp.

```
{
    return id;
}
```

7.13.3.5 nextAction()

```
void KeyStroke::nextAction ( )
```

Definition at line 62 of file keystroke.cpp.

```
{
    id = (id + 1) % action.size();
}
```

7.13.3.6 react()

```
Action * KeyStroke::react ( )
```

Definition at line 44 of file keystroke.cpp.

```
{
    for(auto k : key)
    {
        if(!IsKeyDown(k)) return nullptr;
    }
    return action[id]->clone();
}
```

7.13.3.7 setAction()

```
void KeyStroke::setAction (
             std::vector< Action * > a )
```

Definition at line 34 of file keystroke.cpp.

```
{
    action = a;
}
```

7.13.3.8 size()

```
int KeyStroke::size ( )
```

Definition at line 25 of file keystroke.cpp.

```
{
    return key.size();
}
```

The documentation for this class was generated from the following files:

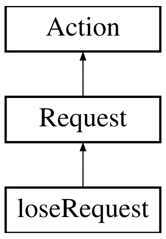
- src/utils/include/keystroke.hpp
- src/utils/src/keystroke.cpp

7.14 loseRequest Class Reference

request sent when the player loses

```
#include <request.hpp>
```

Inheritance diagram for loseRequest:



Public Member Functions

- loseRequest ()=default
- loseRequest (loseRequest ∗)
- ∼loseRequest ()=default
- int isRequest () override
- Action ∗ clone () override
- virtual bool isPackage ()
- virtual void execute ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

Protected Attributes

- ARGS args

7.14.1 Detailed Description

request sent when the player loses

Definition at line 44 of file request.hpp.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 loseRequest() [1/2]

```
loseRequest::loseRequest ( )  [default]
```

7.14.2.2 loseRequest() [2/2]

```
loseRequest::loseRequest (
            loseRequest * other )
```

Definition at line 4 of file lose.cpp.

```
00005 {
00006     args = other->args;
00007 }
```

7.14.2.3 ∼loseRequest()

```
loseRequest::∼loseRequest ( )  [default]
```

7.14.3 Member Function Documentation

7.14.3.1 clone()

```
Action * loseRequest::clone ( )  [override], [virtual]
```

Reimplemented from Request.

Definition at line 14 of file lose.cpp.

```
00015 {
00016     return new loseRequest(this);
00017 }
```

7.14.3.2 execute()

```
void Action::execute ( )  [virtual], [inherited]
```

Reimplemented in CloseAction, resizeAction, PacketAction, moveEntityAction, changeImageAction, moveChunksAction, and moveObjectAction.

Definition at line 25 of file action.cpp.

```
00026 {
00027 }
```

7.14.3.3 getArgs()

```
ARGS & Action::getArgs ( )  [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.14.3.4 isPackage()

```
bool Action::isPackage ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
{
    return false;
}
```

7.14.3.5 isRequest()

```
int loseRequest::isRequest ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 9 of file lose.cpp.

```
{
    return REQUEST::LOSE;
}
```

7.14.3.6 unpack()

```
std::vector< Action * > Action::unpack ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
{
    return std::vector<Action*> ({this});
}
```

7.14.4 Member Data Documentation

7.14.4.1 args

```
ARGS Request::args  [protected], [inherited]
```

Definition at line 17 of file request.hpp.

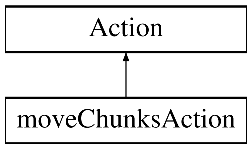
The documentation for this class was generated from the following files:

- src/action/include/request.hpp
- src/action/src/request/lose.cpp

7.15 moveChunksAction Class Reference

```
#include <game.hpp>
```

Inheritance diagram for moveChunksAction:



Public Member Functions

- moveChunksAction (Game ∗, fPoint)
- moveChunksAction (Game ∗, fPoint, float)
- ∼moveChunksAction ()
- void execute () override
- Action ∗ clone () override
- virtual int isRequest ()
- virtual bool isPackage ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

7.15.1 Detailed Description

Definition at line 50 of file game.hpp.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 moveChunksAction() [1/2]

```
moveChunksAction::moveChunksAction (
              Game * game,
              fPoint delta )
```

Definition at line 4 of file movechunk.cpp.

```
{
    this->game = game;
    this->delta = delta;
}
```

7.15.2.2 moveChunksAction() [2/2]

```
moveChunksAction::moveChunksAction (
              Game * game,
              fPoint d,
              float v )
```

Definition at line 10 of file movechunk.cpp.

```
{
    this->game = game;
    this->direction = d;
    this->speed = v;

    float angle = VECTOR2D::getAngle(direction);
    delta[0] = cos(angle) * speed;
    delta[1] = sin(angle) * speed;
}
```

7.15.2.3 ∼moveChunksAction()

```
moveChunksAction::∼moveChunksAction ( )
```

Definition at line 21 of file movechunk.cpp.

```
{
}
```

7.15.3 Member Function Documentation

7.15.3.1 clone()

```
Action * moveChunksAction::clone ( ) [override], [virtual]
```

Reimplemented from Action.

Definition at line 34 of file movechunk.cpp.

```
{
    return new moveChunksAction(game, delta);
}
```

7.15.3.2 execute()

```
void moveChunksAction::execute ( ) [override], [virtual]
```

Reimplemented from Action.

Definition at line 25 of file movechunk.cpp.

```
{
    for(auto& chunk : game->chunks)
    {
        chunk->moveBy(delta);
    }
    game->loadMap();
}
```

7.15.3.3 getArgs()

```
ARGS & Action::getArgs ( ) [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.15.3.4 isPackage()

```
bool Action::isPackage ( ) [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
{
    return false;
}
```

7.15.3.5 isRequest()

```
int Action::isRequest ( )  [virtual], [inherited]
```

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.

```
{
    return 0;
}
```

7.15.3.6 unpack()

```
std::vector< Action * > Action::unpack ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
{
    return std::vector<Action*> ({this});
}
```

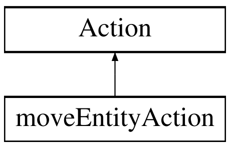
The documentation for this class was generated from the following files:

- src/game/include/game.hpp
- src/game/src/action/movechunk.cpp

7.16 moveEntityAction Class Reference

```
#include <chunk.hpp>
```

Inheritance diagram for moveEntityAction:



Public Member Functions

- moveEntityAction (Chunk ∗)
- ∼moveEntityAction ()
- void execute () override
- Action ∗ clone () override
- virtual int isRequest ()
- virtual bool isPackage ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

7.16.1 Detailed Description

Definition at line 53 of file chunk.hpp.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 moveEntityAction()

```
moveEntityAction::moveEntityAction (
              Chunk * chunk )
```

Definition at line 3 of file moveentity.cpp.

```
00003                                               : chunk(chunk)
00004 {
00005 }
```

7.16.2.2 ∼moveEntityAction()

```
moveEntityAction::∼moveEntityAction ( )
```

Definition at line 7 of file moveentity.cpp.

```
00008 {
00009 }
```

7.16.3 Member Function Documentation

7.16.3.1 clone()

```
Action * moveEntityAction::clone ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 16 of file moveentity.cpp.

```
00017 {
00018     return new moveEntityAction(chunk);
00019 }
```

7.16.3.2 execute()

```
void moveEntityAction::execute ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 11 of file moveentity.cpp.

```
00012 {
00013     chunk->movingEntity();
00014 }
```

7.16.3.3 getArgs()

```
ARGS & Action::getArgs ( ) [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.16.3.4 isPackage()

```
bool Action::isPackage ( ) [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
{
    return false;
}
```

7.16.3.5 isRequest()

```
int Action::isRequest ( ) [virtual], [inherited]
```

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.

```
{
    return 0;
}
```

7.16.3.6 unpack()

```
std::vector< Action * > Action::unpack ( ) [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
{
    return std::vector<Action*> ({this});
}
```

The documentation for this class was generated from the following files:

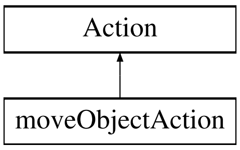
- src/chunk/include/chunk.hpp
- src/chunk/src/action/moveentity.cpp

7.17 moveObjectAction Class Reference

manages the features of a movement, including which object, speed, direction etc.

`#include <interface.hpp>`

Inheritance diagram for moveObjectAction:



Public Member Functions

- moveObjectAction (Container ∗obj, fPoint delta)
- moveObjectAction (Container ∗obj, fPoint dir, float speed)
- ∼moveObjectAction ()
- void execute () override
- Action ∗ clone () override
- virtual int isRequest ()
- virtual bool isPackage ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

7.17.1 Detailed Description

manages the features of a movement, including which object, speed, direction etc.

Definition at line 60 of file interface.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 moveObjectAction() [1/2]

```
moveObjectAction::moveObjectAction (
            Container * obj,
            fPoint delta )
```

Definition at line 4 of file moveobject.cpp.

```
00004                                                                  : obj(obj), delta(delta)
00005 {
00006 }
```

7.17.2.2 moveObjectAction() [2/2]

```
moveObjectAction::moveObjectAction (
            Container * obj,
            fPoint dir,
            float speed )
```

Definition at line 8 of file moveobject.cpp.

```
                                                                        : obj(obj), dir(dir),
      speed(speed)
{
    float angle = VECTOR2D::getAngle(dir);
    delta[0] = cos(angle) * speed;
    delta[1] = sin(angle) * speed;
}
```

7.17.2.3 ∼moveObjectAction()

```
moveObjectAction::∼moveObjectAction ( )
```

Definition at line 15 of file moveobject.cpp.

```
{
}
```

7.17.3 Member Function Documentation

7.17.3.1 clone()

```
Action * moveObjectAction::clone ( ) [override], [virtual]
```

Reimplemented from Action.

Definition at line 25 of file moveobject.cpp.

```
{
    return new moveObjectAction(obj, delta);
}
```

7.17.3.2 execute()

```
void moveObjectAction::execute ( ) [override], [virtual]
```

Reimplemented from Action.

Definition at line 19 of file moveobject.cpp.

```
{
    obj->moveBy(delta);
}
```

7.17.3.3 getArgs()

```
ARGS & Action::getArgs ( ) [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.17.3.4 isPackage()

```
bool Action::isPackage ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.
```
00021 {
00022     return false;
00023 }
```

7.17.3.5 isRequest()

```
int Action::isRequest ( )  [virtual], [inherited]
```

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.
```
00016 {
00017     return 0;
00018 }
```

7.17.3.6 unpack()

```
std::vector< Action * > Action::unpack ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.
```
00035 {
00036     return std::vector<Action*> ({this});
00037 }
```

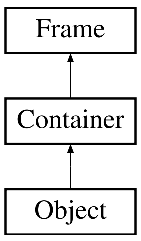
The documentation for this class was generated from the following files:

- src/interface/include/interface.hpp
- src/interface/src/action/moveobject.cpp

7.18 Object Class Reference

```
#include <object.hpp>
```

Inheritance diagram for Object:



Public Member Functions

- Object (Frame ∗, Rectangle)
- Object (Object ∗)
- Object (Object ∗, Rectangle)
- Object (Object ∗, Frame ∗, Rectangle)
- ∼Object ()
- virtual std::string linkContent (std::string) override
- virtual std::string linkContentAbsolute (std::string) override
- virtual Action ∗ react () override
- void draw () override
- std::string getName ()
- void setProbability (int)
- int getProbability ()
- void chooseSprite (int)

  choose a specific sprite from a vector of sprites
- void chooseImage (int)

  choose the state of the sprite
- void chooseImage (int, int)

  choose the state of the sprite
- void nextImage ()

  move to next state of the sprite
- void prevImage ()

  move to previous state of the sprite
- void nextSprite ()

  move to the next sprite
- void prevSprite ()

  move to the previous sprite
- bool isOverlapping (fPoint)
- bool isOverlapping (Rectangle)
- bool isOverlapping (Container ∗)
- float OverlappingArea (Rectangle)
- float OverlappingArea (Container ∗)
- void show ()
- void hide ()
- void toggleVisibility ()
- bool isVisible ()
- int getInstanceId ()
- virtual Action ∗ getRuntimeEvent ()
- void plug (Frame ∗par, fRect rel)

  attach a frame to a parent by relative position
- void plug (Frame ∗par)

  attach a frame to a parent by old relative position
- void unplug ()

  detach a frame from its parent
- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (fPoint rel)
- void resize (int w, int h)

- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- void loadControl (YAML::Node node)
- bool loadName (YAML::Node node)
- void loadSprites (YAML::Node node)
- void loadFocus (YAML::Node node)
- virtual void updateFrame (bool recursive=false)
- bool isroot () const

  return true if this frame is root
- void addSubframe (Frame ∗subframe)

  Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

  Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

7.18.1 Detailed Description

Definition at line 8 of file object.hpp.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Object() [1/4]

```
Object::Object (
            Frame ∗ f,
            Rectangle rel )
```

Definition at line 6 of file constructor.cpp.

```
00006                                      : Container(f, rel)
00007 {
00008     waitUntil = std::chrono::steady_clock::now();
00009 }
```

7.18.2.2 Object() [2/4]

```
Object::Object (
            Object * other )
```

Definition at line 11 of file constructor.cpp.

```
                            : Container(other)
{
    waitUntil = std::chrono::steady_clock::now();
}
```

7.18.2.3 Object() [3/4]

```
Object::Object (
            Object * other,
            Rectangle rel )
```

Definition at line 16 of file constructor.cpp.

```
                                            : Container(other, rel)
{
    waitUntil = std::chrono::steady_clock::now();
}
```

7.18.2.4 Object() [4/4]

```
Object::Object (
            Object * other,
            Frame * f,
            Rectangle rel )
```

Definition at line 21 of file constructor.cpp.

```
                                                     : Container(other, f, rel)
{
    waitUntil = std::chrono::steady_clock::now();
}
```

7.18.2.5 ∼Object()

```
Object::∼Object ( )
```

Definition at line 4 of file destructor.cpp.

```
{
    for (auto &stroke : strokes)
    {
        delete stroke.stroke;
    }
    strokes.clear();

}
```

7.18.3 Member Function Documentation

7.18.3.1 addSubframe()

```
void Frame::addSubframe (
            Frame * subframe )   [protected], [inherited]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.
```
00071 {
00072     mtx.lock();
00073     subframes.push_back(subframe);
00074     mtx.unlock();
00075 }
```

7.18.3.2 beginUpdate()

```
void Frame::beginUpdate ( )  [protected], [inherited]
```

Definition at line 113 of file family.cpp.
```
00114 {
00115     mtx.lock();
00116 }
```

7.18.3.3 chooseImage() [1/2]

```
void Container::chooseImage (
              int index )  [inherited]
```

choose the state of the sprite

Definition at line 231 of file constructor.cpp.
```
00232 {
00233     if(sprites.empty()) return;
00234     if(index < 0 || index >= sprites.size()) return;
00235     focus[1] = index;
00236 }
```

7.18.3.4 chooseImage() [2/2]

```
void Container::chooseImage (
              int index,
              int index2 )  [inherited]
```

choose the state of the sprite

Definition at line 238 of file constructor.cpp.
```
00239 {
00240     if(sprites.empty()) return;
00241     if(index < 0 || index >= sprites.size()) return;
00242     if(index2 < 0 || index2 >= sprites.at(index).size()) return;
00243     focus[0] = index;
00244     focus[1] = index2;
00245 }
```

7.18.3.5 chooseSprite()

```
void Container::chooseSprite (
              int index )  [inherited]
```

choose a specific sprite from a vector of sprites

Definition at line 224 of file constructor.cpp.
```
00225 {
00226     if(sprites.empty()) return;
00227     if(index < 0 || index >= sprites.size()) return;
00228     focus[0] = index;
00229 }
```

7.18.3.6 draw()

```
void Object::draw ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 19 of file arthmetic.cpp.

```
{
    Container::draw();
    return ;
}
```

7.18.3.7 endUpdate()

```
void Frame::endUpdate ( )  [protected], [inherited]
```

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.18.3.8 getCenter()

```
const fPoint & Frame::getCenter ( ) const  [inherited]
```

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.18.3.9 getFrame()

```
const Rectangle & Frame::getFrame ( ) const  [inherited]
```

Definition at line 105 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.18.3.10 getH()

```
const float & Frame::getH ( ) const  [inherited]
```

Definition at line 161 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}
```

7.18.3.11 getInstanceId()

```
int Container::getInstanceId ( )  [inherited]
```

Definition at line 31 of file arthmetic.cpp.

```
00032 {
00033     return instance_id;
00034 }
```

7.18.3.12 getName()

```
std::string Container::getName ( )  [inherited]
```

Definition at line 275 of file constructor.cpp.

```
00276 {
00277     return name;
00278 }
```

7.18.3.13 getParent()

```
Frame * Frame::getParent ( )  [inherited]
```

Definition at line 117 of file arthmetic.cpp.

```
00118 {
00119     std::lock_guard<std::mutex> lock(mtx);
00120     return parent;
00121 }
```

7.18.3.14 getProbability()

```
int Container::getProbability ( )  [inherited]
```

Definition at line 285 of file constructor.cpp.

```
00286 {
00287     return probability;
00288 }
```

7.18.3.15 getRelative()

```
const fRect & Frame::getRelative ( ) const  [inherited]
```

Definition at line 111 of file arthmetic.cpp.

```
00112 {
00113     std::lock_guard<std::mutex> lock(mtx);
00114     return relative;
00115 }
```

7.18.3.16 getRuntimeEvent()

```
Action * Container::getRuntimeEvent ( )  [virtual], [inherited]
```

Reimplemented in Chunk, Game, and Interface.

Definition at line 41 of file arthmetic.cpp.

```
00042 {
00043     return nullptr;
00044 }
```

7.18.3.17 getW()

```
const float & Frame::getW ( ) const  [inherited]
```

Definition at line 155 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}
```

7.18.3.18 getX()

```
const float & Frame::getX ( ) const  [inherited]
```

Definition at line 143 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}
```

7.18.3.19 getY()

```
const float & Frame::getY ( ) const  [inherited]
```

Definition at line 149 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}
```

7.18.3.20 hide()

```
void Container::hide ( )  [inherited]
```

Definition at line 16 of file arthmetic.cpp.

```
{
    visible = false;
}
```

7.18.3.21 isOverlapping() [1/3]

```
bool Container::isOverlapping (
            Container * container )  [inherited]
```

Definition at line 16 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    Rectangle rec2 = container->getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
      rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.18.3.22 isOverlapping() [2/3]

```
bool Container::isOverlapping (
            fPoint point )  [inherited]
```

Definition at line 3 of file overlap.cpp.

```
{
    Rectangle rec = getFrame();
    return (point[0] >= rec.x && point[0] <= rec.x + rec.width && point[1] >= rec.y && point[1] <=
 rec.y + rec.height);

}
```

7.18.3.23 isOverlapping() [3/3]

```
bool Container::isOverlapping (
            Rectangle rec )  [inherited]
```

Definition at line 10 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
 rec2.height && rec.y + rec.height >= rec2.y);
}
```

7.18.3.24 isroot()

```
bool Frame::isroot ( ) const  [protected], [inherited]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.18.3.25 isVisible()

```
bool Container::isVisible ( )  [inherited]
```

Definition at line 26 of file arthmetic.cpp.

```
{
    return visible;
}
```

7.18.3.26 linkContent()

```
std::string Object::linkContent (
            std::string path )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 26 of file constructor.cpp.

```
{
    return linkContentAbsolute(PATB::OBJECT_ + path);
}
```

7.18.3.27 linkContentAbsolute()

```
std::string Object::linkContentAbsolute (
             std::string path )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 31 of file constructor.cpp.

```
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";
    if(node["textures"]) loadSprites(node["textures"]);
    if(node["control"]) loadControl(node["control"]);

    chooseImage(0, 0);
    if(node["focus"])
        loadFocus(node["focus"]);

    return "";
}
```

7.18.3.28 loadControl()

```
void Object::loadControl (
             YAML::Node node )  [protected]
```

Definition at line 45 of file constructor.cpp.

```
{
    for(auto stroke : node)
    {
        strokes.emplace_back();
        KeyStroke* k = new KeyStroke();
        for(auto key : stroke["key"])
        {
            k->add(toKey(key.as<std::string>()));
        }
        for(auto sprite : stroke["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            k->addAction(new changeImageAction(this, p));
        }
        strokes.back().stroke = k;
    }
}
```

7.18.3.29 loadFocus()

```
void Container::loadFocus (
             YAML::Node node )  [protected], [inherited]
```

Definition at line 218 of file constructor.cpp.

```
{
    focus[0] = node[0].as<int>();
    focus[1] = node[1].as<int>();
}
```

7.18.3.30 loadName()

```
bool Container::loadName (
            YAML::Node node ) [protected], [inherited]
```

Definition at line 111 of file constructor.cpp.

```
{
    if(!node["name"])
    {
        name = "";
        return false;
    }
    name = node["name"].as<std::string>();
    return true;
}
```

7.18.3.31 loadSprites()

```
void Container::loadSprites (
            YAML::Node node ) [protected], [inherited]
```

Definition at line 122 of file constructor.cpp.

```
{
    for(auto sprite : node)
    {
        if(!sprite["path"]) continue;
        if(!sprite["graphics"]) continue;

        std::string path = PASSETS::GRAPHIC_ + sprite["path"].as<std::string>();
        Image image = LoadImage(path.c_str());

        if(sprite["resize"])
        {
            int x = image.width * sprite["resize"][0].as<float>();
            int y = image.height * sprite["resize"][1].as<float>();
            ImageResize(&image, x, y);
        }

        sprites.emplace_back();
        for(auto img : sprite["graphics"])
        {
            float x, y, w, h;
            int repeat = 1;
            int gapX = 0;
            int gapY = 0;

            int dx = 1;
            int dy = 1;

            if(img["x"])
                x = img["x"].as<float>() / 100.0;
            else x = 0;
            if(img["y"])
                y = img["y"].as<float>() / 100.0;
            else y = 0;
            if(img["w"])
                w = img["w"].as<float>() / 100.0;
            else w = 1;
            if(img["h"])
                h = img["h"].as<float>() / 100.0;
            else h = 1;
            if(img["repeat"])
                repeat = img["repeat"].as<int>();
            if(img["gapX"])
                gapX = img["gapX"].as<int>();
            if(img["gapY"])
                gapY = img["gapY"].as<int>();

            if(img["dx"])
                dx = img["dx"].as<int>();
            if(dx < 0) dx = -1;
            else dx = 1;

            if(img["dy"])
                dy = img["dy"].as<int>();
            if(dy < 0) dy = -1;
            else dy = 1;

            int imgw = image.width;
            int imgh = image.height;

            if(img["axis"] && img["axis"].as<std::string>() == "horizontal")
            {
                for(float j = y; j >= 0 && j + h < 1 + 1e-2; j += dy * (gapY + h))
                {
                    for(float i = x; i >= 0 && i + w <= 1 + 1e-2 && repeat--; i += dx * (gapX + w))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }else
            {
                for(float i = x; i >= 0 && i + w <= 1 + 1e-2; i += dx * (gapX + w))
                {
                    for(float j = y; j >= 0 && j + h < 1 + 1e-2 && repeat--; j += dy * (gapY + h))
                    {
                        Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                        Image img2 = ImageFromImage(image, rect);
                        Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                        Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                        sprites.back().push_back(vis);

                        UnloadImage(img2);
                    }
                }
            }
        }
        UnloadImage(image);
    }
}
```

7.18.3.32 moveBy() [1/2]

```
void Frame::moveBy (
            fPoint rel )   [inherited]
```

Definition at line 65 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.33 moveBy() [2/2]

```
void Frame::moveBy (
            int x,
            int y )   [inherited]
```

Definition at line 75 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.34 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
            fPoint rel )  [inherited]
```

Definition at line 43 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.35 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
            int x,
            int y )  [inherited]
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.36 moveTo() [1/2]

```
void Frame::moveTo (
            fPoint rel )  [inherited]
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.37 moveTo() [2/2]

```
void Frame::moveTo (
            int x,
            int y )  [inherited]
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.38 nextImage()

```
void Container::nextImage ( )  [inherited]
```

move to next state of the sprite

Definition at line 247 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]++;
    if(focus[1] >= sprites.at(focus[0]).size()) focus[1] = 0;
}
```

7.18.3.39 nextSprite()

```
void Container::nextSprite ( )  [inherited]
```

move to the next sprite

Definition at line 261 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]++;
    if(focus[0] >= sprites.size()) focus[0] = 0;
}
```

7.18.3.40 operator fRect()

```
Frame::operator fRect ( ) const  [inherited]
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.18.3.41 operator iRect()

```
Frame::operator iRect ( ) const  [inherited]
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.18.3.42 operator Rectangle()

```
Frame::operator Rectangle ( ) const  [inherited]
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.18.3.43 OverlappingArea() [1/2]

```
float Container::OverlappingArea (
            Container * container )  [inherited]
```

Definition at line 34 of file overlap.cpp.

```
{
    Rectangle rec = container->getFrame();
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.18.3.44 OverlappingArea() [2/2]

```
float Container::OverlappingArea (
            Rectangle rec )  [inherited]
```

Definition at line 23 of file overlap.cpp.

```
{
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

7.18.3.45 plug() [1/2]

```
void Frame::plug (
            Frame * par )  [inherited]
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.18.3.46 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )  [inherited]
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.18.3.47 prevImage()

```
void Container::prevImage ( )  [inherited]
```

move to previous state of the sprite

Definition at line 254 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[1]--;
    if(focus[1] < 0) focus[1] = sprites.at(focus[0]).size() - 1;
}
```

7.18.3.48 prevSprite()

```
void Container::prevSprite ( )  [inherited]
```

move to the previous sprite

Definition at line 268 of file constructor.cpp.

```
{
    if(sprites.empty()) return;
    focus[0]--;
    if(focus[0] < 0) focus[0] = sprites.size() - 1;
}
```

7.18.3.49 react()

```
Action * Object::react ( )  [override], [virtual]
```

Reimplemented from Container.

Definition at line 3 of file arthmetic.cpp.

```
{
    if(std::chrono::steady_clock::now() < waitUntil)
        return nullptr;
    for(int i = 0; i < strokes.size(); i++)
    {
        Action* a = strokes[i].stroke->react();
        if(a == nullptr) continue;
        else strokes[i].stroke->nextAction();
        return a;
    }


    return nullptr;
}
```

7.18.3.50 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe )  [protected], [inherited]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.

```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.18.3.51 resize() [1/2]

```
void Frame::resize (
            fPoint rel )  [inherited]
```

Definition at line 85 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.52 resize() [2/2]

```
void Frame::resize (
            int w,
            int h )  [inherited]
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.53 setProbability()

```
void Container::setProbability (
              int prob )  [inherited]
```

Definition at line 280 of file constructor.cpp.

```
{
    probability = prob;
}
```

7.18.3.54 setRelative()

```
void Frame::setRelative (
              fRect rel )  [inherited]
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.18.3.55 show()

```
void Container::show ( )  [inherited]
```

Definition at line 11 of file arthmetic.cpp.

```
{
    visible = true;
}
```

7.18.3.56 toggleVisibility()

```
void Container::toggleVisibility ( )  [inherited]
```

Definition at line 21 of file arthmetic.cpp.

```
{
    visible = !visible;
}
```

7.18.3.57 unplug()

```
void Frame::unplug ( )  [inherited]
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.18.3.58 updateFrame()

```
void Frame::updateFrame (
            bool recursive = false )  [protected], [virtual], [inherited]
```

Reimplemented in Visual.

Definition at line 3 of file arthmetic.cpp.

```
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}
```

The documentation for this class was generated from the following files:

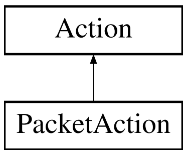
- src/object/include/object.hpp
- src/object/src/arthmetic.cpp
- src/object/src/constructor.cpp
- src/object/src/destructor.cpp

7.19 PacketAction Class Reference

organize selected actions into a package

```
#include <action.hpp>
```

Inheritance diagram for PacketAction:



Public Member Functions

- PacketAction ()
- PacketAction (PacketAction ∗)
- ∼PacketAction ()
- bool isPackage () override
- void addAction (Action ∗)
- void addAction (PacketAction ∗)
- std::vector< Action ∗ > unpack () override
- void execute () override
- PacketAction ∗ clone () override
- virtual int isRequest ()
- virtual ARGS & getArgs ()

7.19.1 Detailed Description

organize selected actions into a package

Definition at line 52 of file action.hpp.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 PacketAction() [1/2]

```
PacketAction::PacketAction ( )
```

Definition at line 44 of file action.cpp.

```
00044                               : Action()
00045 {
00046 }
```

7.19.2.2 PacketAction() [2/2]

```
PacketAction::PacketAction (
               PacketAction * action )
```

Definition at line 49 of file action.cpp.

```
00049                                                   : Action(action)
00050 {
00051     for(Action* a : action->actions)
00052         actions.push_back(a->clone());
00053 }
```

7.19.2.3 ∼PacketAction()

```
PacketAction::∼PacketAction ( )
```

Definition at line 55 of file action.cpp.

```
00056 {
00057     for(Action* a : actions)
00058         delete a;
00059     actions.clear();
00060 }
```

7.19.3 Member Function Documentation

7.19.3.1 addAction() [1/2]

```
void PacketAction::addAction (
               Action * action )
```

Definition at line 67 of file action.cpp.

```
00068 {
00069     actions.push_back(action);
00070 }
```

7.19.3.2 addAction() [2/2]

```
void PacketAction::addAction (
            PacketAction * action )
```

Definition at line 72 of file action.cpp.

```
{
    for(auto i : action->actions)
        actions.push_back(i);
    action->actions.clear();
}
```

7.19.3.3 clone()

```
PacketAction * PacketAction::clone ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 116 of file action.cpp.

```
{
    return new PacketAction(this);
}
```

7.19.3.4 execute()

```
void PacketAction::execute ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 107 of file action.cpp.

```
{
    for(Action* a : actions)
    {
        a->execute();
    }
}
```

7.19.3.5 getArgs()

```
ARGS & Action::getArgs ( )  [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.19.3.6 isPackage()

```
bool PacketAction::isPackage ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 62 of file action.cpp.

```
{
    return true;
}
```

7.19.3.7 isRequest()

```
int Action::isRequest ( )  [virtual], [inherited]
```

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.

```
{
    return 0;
}
```

7.19.3.8 unpack()

```
std::vector< Action * > PacketAction::unpack ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 79 of file action.cpp.

```
{
    std::vector<Action*> unpacked;
    std::queue<PacketAction*> q;

    q.push(this);

    while(!q.empty())
    {
        PacketAction* p = q.front();
        q.pop();

        for(Action* a : p->actions)
        {
            if(a->isPackage())
            {
                q.push((PacketAction*)a);
            }
            else
            {
                unpacked.push_back(a);
            }
        }
        p->actions.clear();
    }
    return unpacked;
}
```

The documentation for this class was generated from the following files:

- src/action/include/action.hpp
- src/action/src/action.cpp

7.20 RandomEngine Class Reference

```
#include <random.hpp>
```

Public Member Functions

- RandomEngine ()
- RandomEngine (unsigned int seed)
- ∼RandomEngine ()
- int randInt (int min=0, int max=1)
- double randDouble (double min=0, double max=1)
- char randChar (char min=0, char max=127)
- std::string randString (int length, char min, char max)
- std::string randInt2String (int length, int min=0, int max=9)
- std::string randString (int length, bool haveDigit=true, bool haveLower=true, bool haveUpper=true, bool haveSpecial=true)

7.20.1 Detailed Description

Definition at line 7 of file random.hpp.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 RandomEngine() [1/2]

```
RandomEngine::RandomEngine ( )
```

Definition at line 6 of file random.cpp.

```
{
    unsigned int seed = std::chrono::system_clock::now().time_since_epoch().count();
    engine.seed(seed);
}
```

7.20.2.2 RandomEngine() [2/2]

```
RandomEngine::RandomEngine (
            unsigned int seed )
```

Definition at line 12 of file random.cpp.

```
{
    engine.seed(seed);
}
```

7.20.2.3 ∼RandomEngine()

```
RandomEngine::∼RandomEngine ( )
```

Definition at line 17 of file random.cpp.

```
{
}
```

7.20.3 Member Function Documentation

7.20.3.1 randChar()

```
char RandomEngine::randChar (
            char min = 0,
            char max = 127 )
```

Definition at line 33 of file random.cpp.

```
{
    std::uniform_int_distribution<int> distribution(min, max);
    return distribution(engine);
}
```

7.20.3.2 randDouble()

```
double RandomEngine::randDouble (
              double min = 0,
              double max = 1 )
```

Definition at line 27 of file random.cpp.

```
{
    std::uniform_real_distribution<double> distribution(min, max);
    return distribution(engine);
}
```

7.20.3.3 randInt()

```
int RandomEngine::randInt (
              int min = 0,
              int max = 1 )
```

Definition at line 21 of file random.cpp.

```
{
    std::uniform_int_distribution<int> distribution(min, max);
    return distribution(engine);
}
```

7.20.3.4 randInt2String()

```
std::string RandomEngine::randInt2String (
              int length,
              int min = 0,
              int max = 9 )
```

Definition at line 49 of file random.cpp.

```
{
    std::string str;
    for (int i = 0; i < length; i++)
    {
        str += std::to_string(randInt(min, max));
    }
    return str;
}
```

7.20.3.5 randString() [1/2]

```
std::string RandomEngine::randString (
              int length,
              bool haveDigit = true,
              bool haveLower = true,
              bool haveUpper = true,
              bool haveSpecial = true )
```

Definition at line 59 of file random.cpp.

```
{
    std::string str;
    std::string digit = "0123456789";
    std::string lower = "abcdefghijklmnopqrstuvwxyz";
    std::string upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    std::string special = "!@#$%^&*()_+{}|:<>?~`-=[]\\;',./\"";
    std::string all = digit + lower + upper + special;
    if (haveDigit)
    {
        str += digit;
    }
    if (haveLower)
    {
        str += lower;
    }
    if (haveUpper)
    {
        str += upper;
    }
    if (haveSpecial)
    {
        str += special;
    }
    if (str.empty())
    {
        str = all;
    }
    std::string result;
    for (int i = 0; i < length; i++)
    {
        result += str[randInt(0, str.size() - 1)];
    }
    return result;
}
```

7.20.3.6 randString() `[2/2]`

```
std::string RandomEngine::randString (
              int length,
              char min,
              char max )
```

Definition at line 39 of file random.cpp.

```
{
    std::string str;
    for (int i = 0; i < length; i++)
    {
        str += randChar(min, max);
    }
    return str;
}
```

The documentation for this class was generated from the following files:

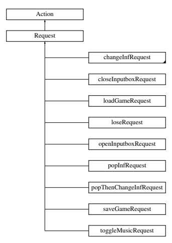
- src/utils/include/random.hpp
- src/utils/src/random.cpp

7.21 Request Class Reference

sends information to a higher, relevant entity

```
#include <request.hpp>
```

Inheritance diagram for Request:



Public Member Functions

- Request ()
- Request (Request ∗)
- ∼Request ()=default
- int isRequest () override
- virtual Action ∗ clone () override
- virtual bool isPackage ()
- virtual void execute ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

Protected Attributes

- ARGS args

7.21.1 Detailed Description

sends information to a higher, relevant entity

upgrades information to higher level

Definition at line 14 of file request.hpp.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Request() [1/2]

```
Request::Request ( )
```

Definition at line 4 of file request.cpp.

```
00004                   : Action()
00005 {
00006 }
```

7.21.2.2 Request() [2/2]

```
Request::Request (
              Request ∗ request )
```

Definition at line 8 of file request.cpp.

```
00008                                   : Action(request)
00009 {
00010 }
```

7.21.2.3 ∼Request()

```
Request::∼Request ( )  [default]
```

7.21.3 Member Function Documentation

7.21.3.1 clone()

```
Action * Request::clone ( )  [override], [virtual]
```

Reimplemented from Action.

Reimplemented in changeInfRequest, and loseRequest.

Definition at line 17 of file request.cpp.

```
{
    return new Request(this);
}
```

7.21.3.2 execute()

```
void Action::execute ( )  [virtual], [inherited]
```

Reimplemented in CloseAction, resizeAction, PacketAction, moveEntityAction, changeImageAction, moveChunksAction, and moveObjectAction.

Definition at line 25 of file action.cpp.

```
{
}
```

7.21.3.3 getArgs()

```
ARGS & Action::getArgs ( )  [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
{
    return NONE_ARGS;
}
```

7.21.3.4 isPackage()

```
bool Action::isPackage ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
{
    return false;
}
```

7.21.3.5 isRequest()

```
int Request::isRequest ( )  [override], [virtual]
```

Reimplemented from Action.

Definition at line 12 of file request.cpp.

```
{
    return 1;
}
```

7.21.3.6 unpack()

```
std::vector< Action * > Action::unpack ( ) [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
00035 {
00036     return std::vector<Action*> ({this});
00037 }
```

7.21.4 Member Data Documentation

7.21.4.1 args

```
ARGS Request::args [protected]
```

Definition at line 17 of file request.hpp.

The documentation for this class was generated from the following files:

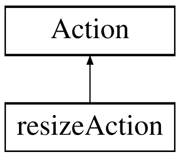
- src/action/include/request.hpp
- src/action/src/request.cpp

7.22 resizeAction Class Reference

manages the resizing of the window

```
#include <window.hpp>
```

Inheritance diagram for resizeAction:



Public Member Functions

- resizeAction (Window ∗window, float w, float h)
- ∼resizeAction ()=default
- void execute ()
- virtual int isRequest ()
- virtual bool isPackage ()
- virtual Action ∗ clone ()
- virtual std::vector< Action ∗ > unpack ()
- virtual ARGS & getArgs ()

7.22.1 Detailed Description

manages the resizing of the window

Definition at line 197 of file window.hpp.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 resizeAction()

```
resizeAction::resizeAction (
            Window * window,
            float w,
            float h )
```

Definition at line 3 of file resize.cpp.

```
00004 {
00005     win = window;
00006     w = x;
00007     h = y;
00008 }
```

7.22.2.2 ∼resizeAction()

```
resizeAction::∼resizeAction ( )  [default]
```

7.22.3 Member Function Documentation

7.22.3.1 clone()

```
Action * Action::clone ( )  [virtual], [inherited]
```

Reimplemented in PacketAction, Request, changeInfRequest, loseRequest, moveEntityAction, changeImageAction, moveChunksAction, and moveObjectAction.

Definition at line 29 of file action.cpp.

```
00030 {
00031     return this;
00032 }
```

7.22.3.2 execute()

```
void resizeAction::execute ( )  [virtual]
```

Reimplemented from Action.

Definition at line 10 of file resize.cpp.

```
00011 {
00012     win->UI.resize(w, h);
00013 }
```

7.22.3.3 getArgs()

```
ARGS & Action::getArgs ( )  [virtual], [inherited]
```

Reimplemented in changeInfRequest.

Definition at line 39 of file action.cpp.

```
00040 {
00041     return NONE_ARGS;
00042 }
```

7.22.3.4 isPackage()

```
bool Action::isPackage ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 20 of file action.cpp.

```
00021 {
00022     return false;
00023 }
```

7.22.3.5 isRequest()

```
int Action::isRequest ( )  [virtual], [inherited]
```

Reimplemented in Request, changeInfRequest, and loseRequest.

Definition at line 15 of file action.cpp.

```
00016 {
00017     return 0;
00018 }
```

7.22.3.6 unpack()

```
std::vector< Action * > Action::unpack ( )  [virtual], [inherited]
```

Reimplemented in PacketAction.

Definition at line 34 of file action.cpp.

```
00035 {
00036     return std::vector<Action*> ({this});
00037 }
```

The documentation for this class was generated from the following files:

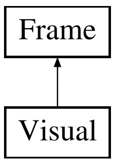
- src/window/include/window.hpp
- src/window/src/action/resize.cpp

7.23 Visual Class Reference

images displayed on screen

```
#include <visual.hpp>
```

Inheritance diagram for Visual:



Public Member Functions

- Visual (Texture2D ∗, Frame ∗, Rectangle)
- Visual (Visual ∗)
- Visual (Visual ∗, Rectangle)
- Visual (Visual ∗, Frame ∗, Rectangle)
- ∼Visual ()
- void resize (fPoint)
- void draw ()
- void plug (Frame ∗par, fRect rel)

  attach a frame to a parent by relative position
- void plug (Frame ∗par)

  attach a frame to a parent by old relative position
- void unplug ()

  detach a frame from its parent
- void moveTo (fPoint rel)
- void moveTo (int x, int y)
- void moveCenterTo (fPoint rel)
- void moveCenterTo (int x, int y)
- void moveBy (fPoint rel)
- void moveBy (int, int)
- void resize (int w, int h)
- const Rectangle & getFrame () const
- const fRect & getRelative () const
- Frame ∗ getParent ()
- void setRelative (fRect rel)
- const fPoint & getCenter () const
- const float & getX () const
- const float & getY () const
- const float & getW () const
- const float & getH () const
- operator Rectangle () const
- operator fRect () const
- operator iRect () const

Protected Member Functions

- void fitFrame ()
- void updateFrame (bool recursive=false) override
- bool isroot () const

    return true if this frame is root
- void addSubframe (Frame ∗subframe)

    Add a subframe to this frame.
- void removeSubframe (Frame ∗subframe)

    Remove a subframe from this frame.
- void beginUpdate ()
- void endUpdate ()

7.23.1 Detailed Description

images displayed on screen

Definition at line 16 of file visual.hpp.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Visual() [1/4]

```
Visual::Visual (
            Texture2D * txtr,
            Frame * frame,
            Rectangle rect )
```

Definition at line 4 of file constructor.cpp.

```
                                                      : Frame(frame, rect)
{
    m_texture = std::shared_ptr<Texture2D>(txtr, [](Texture2D* texture){
        UnloadTexture(*texture);
        texture = nullptr;
    });
    resizeable = true;
    fitFrame();
}
```

7.23.2.2 Visual() [2/4]

```
Visual::Visual (
            Visual * visual )
```

Definition at line 14 of file constructor.cpp.

```
                                 : Frame(visual)
{
    resizeable = false;
    m_texture = visual->m_texture;
    fitFrame();
}
```

7.23.2.3 Visual() [3/4]

```
Visual::Visual (
            Visual * visual,
            Rectangle rect )
```

Definition at line 21 of file constructor.cpp.

```
00021                                          : Frame(visual, rect)
00022 {
00023     resizeable = false;
00024     m_texture = visual->m_texture;
00025     fitFrame();
00026 }
```

7.23.2.4 Visual() [4/4]

```
Visual::Visual (
            Visual * visual,
            Frame * frame,
            Rectangle rect )
```

Definition at line 28 of file constructor.cpp.

```
00028                                                      : Frame(frame, rect)
00029 {
00030     resizeable = false;
00031     m_texture = visual->m_texture;
00032     fitFrame();
00033 }
```

7.23.2.5 ∼Visual()

```
Visual::∼Visual ( )
```

Definition at line 12 of file destructor.cpp.

```
00013 {
00014     m_texture.reset();
00015 }
```

7.23.3 Member Function Documentation

7.23.3.1 addSubframe()

```
void Frame::addSubframe (
            Frame * subframe )  [protected], [inherited]
```

Add a subframe to this frame.

When unplug a subframe, parent frame will call this function, so you shouldn't call it

Parameters

subframe	subframe to add

Definition at line 70 of file family.cpp.

```
00071 {
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}
```

7.23.3.2 beginUpdate()

`void Frame::beginUpdate ( )  [protected], [inherited]`

Definition at line 113 of file family.cpp.

```
{
    mtx.lock();
}
```

7.23.3.3 draw()

`void Visual::draw ( )`

Definition at line 4 of file arthmetic.cpp.

```
{
    if(m_texture == nullptr) return ;
    Rectangle rec = getFrame();
    // draw texture
    DrawTexture(*m_texture, rec.x, rec.y, WHITE);
}
```

7.23.3.4 endUpdate()

`void Frame::endUpdate ( )  [protected], [inherited]`

Definition at line 118 of file family.cpp.

```
{
    mtx.unlock();
}
```

7.23.3.5 fitFrame()

`void Visual::fitFrame ( )  [protected]`

Definition at line 12 of file arthmetic.cpp.

```
{
    if(m_texture == nullptr) return ;
    if(!resizeable) return ;
    const Rectangle &rec = Frame::getFrame();

    Image img = LoadImageFromTexture(*m_texture);
    UnloadTexture(*m_texture.get());

    ImageResize(&img, rec.width, rec.height);
    *m_texture.get() = LoadTextureFromImage(img);
    UnloadImage(img);
}
```

7.23.3.6 getCenter()

`const fPoint & Frame::getCenter ( ) const  [inherited]`

Definition at line 131 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}
```

7.23.3.7 getFrame()

```
const Rectangle & Frame::getFrame ( ) const  [inherited]
```

Definition at line 105 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.23.3.8 getH()

```
const float & Frame::getH ( ) const  [inherited]
```

Definition at line 161 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}
```

7.23.3.9 getParent()

```
Frame * Frame::getParent ( )  [inherited]
```

Definition at line 117 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent;
}
```

7.23.3.10 getRelative()

```
const fRect & Frame::getRelative ( ) const  [inherited]
```

Definition at line 111 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.23.3.11 getW()

```
const float & Frame::getW ( ) const  [inherited]
```

Definition at line 155 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}
```

7.23.3.12 getX()

```
const float & Frame::getX ( ) const  [inherited]
```

Definition at line 143 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}
```

7.23.3.13 getY()

```
const float & Frame::getY ( ) const  [inherited]
```

Definition at line 149 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}
```

7.23.3.14 isroot()

```
bool Frame::isroot ( ) const  [protected], [inherited]
```

return true if this frame is root

Definition at line 107 of file family.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}
```

7.23.3.15 moveBy() [1/2]

```
void Frame::moveBy (
             fPoint rel )  [inherited]
```

Definition at line 65 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.16 moveBy() [2/2]

```
void Frame::moveBy (
             int x,
             int y )  [inherited]
```

Definition at line 75 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.17 moveCenterTo() [1/2]

```
void Frame::moveCenterTo (
             fPoint rel )  [inherited]
```

Definition at line 43 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.18 moveCenterTo() [2/2]

```
void Frame::moveCenterTo (
            int x,
            int y )  [inherited]
```

Definition at line 54 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.19 moveTo() [1/2]

```
void Frame::moveTo (
            fPoint rel )  [inherited]
```

Definition at line 24 of file arthmetic.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.20 moveTo() [2/2]

```
void Frame::moveTo (
            int x,
            int y )  [inherited]
```

Definition at line 33 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.21 operator fRect()

```
Frame::operator fRect ( ) const  [inherited]
```

Definition at line 173 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}
```

7.23.3.22 operator iRect()

```
Frame::operator iRect ( ) const  [inherited]
```

Definition at line 179 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}
```

7.23.3.23 operator Rectangle()

```
Frame::operator Rectangle ( ) const  [inherited]
```

Definition at line 167 of file arthmetic.cpp.

```
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}
```

7.23.3.24 plug() [1/2]

```
void Frame::plug (
            Frame * par )  [inherited]
```

attach a frame to a parent by old relative position

Parameters

par	parent frame

Definition at line 34 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.23.3.25 plug() [2/2]

```
void Frame::plug (
            Frame * par,
            fRect rel )  [inherited]
```

attach a frame to a parent by relative position

Parameters

par	parent frame
rel	relative position and size in percentage (0.0f to 1.0f)

Definition at line 12 of file family.cpp.

```
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}
```

7.23.3.26 removeSubframe()

```
void Frame::removeSubframe (
            Frame * subframe )  [protected], [inherited]
```

Remove a subframe from this frame.

When destroy a subframe that have parent frame, this function is called, so you shouldn't call it

Parameters

subframe	subframe to remove

Definition at line 85 of file family.cpp.

```
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}
```

7.23.3.27 resize() [1/2]

```
void Visual::resize (
            fPoint rel )
```

Definition at line 26 of file arthmetic.cpp.

```
{
    Frame::resize(rel);
    updateFrame(true);
}
```

7.23.3.28 resize() [2/2]

```
void Frame::resize (
            int w,
            int h )  [inherited]
```

Definition at line 95 of file arthmetic.cpp.

```
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.29 setRelative()

```
void Frame::setRelative (
             fRect rel )  [inherited]
```

Definition at line 123 of file arthmetic.cpp.

```
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}
```

7.23.3.30 unplug()

```
void Frame::unplug ( )  [inherited]
```

detach a frame from its parent

Definition at line 53 of file family.cpp.

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}
```

7.23.3.31 updateFrame()

```
void Visual::updateFrame (
             bool recursive = false )  [override], [protected], [virtual]
```

Reimplemented from Frame.

Definition at line 32 of file arthmetic.cpp.

```
{
    if(m_texture == nullptr) return ;
    float prx = getFrame().width;
    float pry = getFrame().height;
    Frame::updateFrame(recursive);

    float rx = getFrame().width - prx;
    float ry = getFrame().height - pry;
    if(rx < 1e-3 && ry < 1e-3) return ;
    fitFrame();
}
```

The documentation for this class was generated from the following files:

- src/visual/include/visual.hpp
- src/visual/src/arthmetic.cpp
- src/visual/src/constructor.cpp
- src/visual/src/destructor.cpp

7.24 Window Class Reference

```
#include <window.hpp>
```

Public Member Functions

- Window ()
- Window (std::string path)
- ∼Window ()
- bool isRun ()
- bool isClose ()
- void run ()

Protected Member Functions

- void draw ()
- void systemEvent ()
- void getUserEvent ()
- void getRuntimeEvent ()
- void sound_effect ()
- void immediateActing ()
- void userActing ()
- void requestActing ()
- void systemActing ()
- void initRaylib (YAML::Node node)
- void loadInterface (YAML::Node node)
- void loadGame (YAML::Node node)

Friends

- class CloseAction
- class resizeAction

7.24.1 Detailed Description

Definition at line 21 of file window.hpp.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 Window() [1/2]

```
Window::Window ( )
```

Definition at line 7 of file constructor.cpp.

```
{
    Wcontent.width = 1200;
    Wcontent.height = 668;
    Wcontent.title = "Crossy Road clone";
    SetConfigFlags(FLAG_WINDOW_RESIZABLE | FLAG_VSYNC_HINT);
    InitWindow(Wcontent.width, Wcontent.height, Wcontent.title.c_str());
    SetTargetFPS(60);

    UI.setRootFrame(new Frame({0, 0, Wcontent.width, Wcontent.height}));
}
```

7.24.2.2 Window() [2/2]

```
Window::Window (
            std::string path )
```

Definition at line 19 of file constructor.cpp.

```
{
    path = PATB::WINDOW_ + path;
    YAML::Node config = YAML_FILE::readFile(path);
    initRaylib(config);

    loadInterface(config["interface-list"]);
    loadGame(config["game"]);
    if(config["choose-interface"])
        UI.push(config["choose-interface"].as<std::string>());

    if(config["input-delay"])
    {
        double delay = config["input-delay"].as<int>() / 1000.0;
        Wcontent.input_delay = std::chrono::duration<double>(delay);
    }else
    {
        Wcontent.input_delay = std::chrono::duration<int>(50) / 1000.0;
    }

    if(config["runtime-delay"])
    {
        double delay = config["runtime-delay"].as<int>() / 1000.0;
        Wcontent.runtime_delay = std::chrono::duration<double>(delay);
    }else
    {
        Wcontent.runtime_delay = std::chrono::duration<int>(40) / 1000.0;
    }
}
```

7.24.2.3 ∼Window()

```
Window::∼Window ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    CloseWindow();
}
```

7.24.3 Member Function Documentation

7.24.3.1 draw()

```
void Window::draw ( )  [protected]
```

Definition at line 34 of file running.cpp.

```
{
    {
        BeginDrawing();
        UI.draw();
        EndDrawing();
    }

}
```

7.24.3.2 getRuntimeEvent()

```
void Window::getRuntimeEvent ( )  [protected]
```

Definition at line 97 of file running.cpp.

```
{
    while(isRun())
    {

        if(!Wcontent.isRuntimeDelayOver())
        {
            std::this_thread::sleep_for(std::chrono::milliseconds(10));
            continue;
        }
        Action* action = UI.getRuntimeEvent();
        if(action != nullptr)
        {
            if(action->isPackage())
            {
                for(auto act : action->unpack())
                {
                    if(act->isRequest())
                        request_pool.push(act);
                    else
                        immediate_user_pool.push(act);
                }
            }
            else if (!action->isRequest())
                immediate_pool.push(action);
        }
        Wcontent.setRuntimeClock2Now();
    }
}
```

7.24.3.3 getUserEvent()

```
void Window::getUserEvent ( )  [protected]
```

Definition at line 66 of file running.cpp.

```
{
    while(isRun())
    {
        if(!Wcontent.isInputDelayOver())
        {
            std::this_thread::sleep_for(std::chrono::milliseconds(10));
            continue;
        }

        Action* action = UI.react();
        if(action != nullptr)
        {
            if(action->isPackage())
            {
                for(auto act : action->unpack())
                {
                    if(act->isRequest())
                        request_pool.push(act);
                    else
                        immediate_user_pool.push(act);
                }
            }
            else if(!action->isRequest())
                immediate_user_pool.push(action);
        }

        Wcontent.setInputClock2Now();
    }
}
```

7.24.3.4 immediateActing()

```
void Window::immediateActing ( )  [protected]
```

Definition at line 50 of file acting.cpp.

```
{
    while(isRun())
    {
        Action* action = immediate_pool.pop();
        if(action == nullptr) continue;
        if(!isRun()) break;
        action->execute();
        delete action;
    }
}
```

7.24.3.5 initRaylib()

```
void Window::initRaylib (
                YAML::Node node )   [protected]
```

Definition at line 50 of file constructor.cpp.

```
{
    Wcontent.width = config["width"].as<int>();
    Wcontent.height = config["height"].as<int>();
    Wcontent.title = config["title"].as<std::string>();
    // enable resizeable window and vsync

    SetConfigFlags(FLAG_WINDOW_RESIZABLE | FLAG_VSYNC_HINT);
    InitWindow(Wcontent.width, Wcontent.height, Wcontent.title.c_str());
    SetTargetFPS(60);
    Wcontent.setStatus(true);
    UI.setRootFrame(new Frame({0, 0, Wcontent.width, Wcontent.height}));
}
```

7.24.3.6 isClose()

```
bool Window::isClose ( )
```

Definition at line 136 of file running.cpp.

```
{
    return !Wcontent.getStatus();
}
```

7.24.3.7 isRun()

```
bool Window::isRun ( )
```

Definition at line 131 of file running.cpp.

```
{
    return Wcontent.getStatus();
}
```

7.24.3.8 loadGame()

```
void Window::loadGame (
                YAML::Node node )   [protected]
```

Definition at line 86 of file constructor.cpp.

```
{
    if(!node) return ;

    std::string path = node["file"].as<std::string>();
    float x = 0, y = 0, w = 1, h = 1;

    if(node["x"]) x = node["x"].as<float>() / 100;
    if(node["y"]) y = node["y"].as<float>() / 100;
    if(node["w"]) w = node["w"].as<float>() / 100;
    if(node["h"]) h = node["h"].as<float>() / 100;


    Interface* inf = new Game(UI.getRootFrame(), {x, y, w, h});
    inf->linkContent(path);
    UI.load(inf);
}
```

7.24.3.9 loadInterface()

```
void Window::loadInterface (
             YAML::Node node )  [protected]
```

Definition at line 64 of file constructor.cpp.

```
{
    if(!node) return ;

    UI.setInterfacePool(new InterfacePool());
    for(auto i : node)
    {
        std::string path = i["file"].as<std::string>();
        float x = 0, y = 0, w = 1, h = 1;

        if(i["x"]) x = i["x"].as<float>() / 100;
        if(i["y"]) y = i["y"].as<float>() / 100;
        if(i["w"]) w = i["w"].as<float>() / 100;
        if(i["h"]) h = i["h"].as<float>() / 100;


        Interface* inf = new Interface(UI.getRootFrame(), {x, y, w, h});
        inf->linkContent(path);
        UI.load(inf);
    }
}
```

7.24.3.10 requestActing()

```
void Window::requestActing ( )  [protected]
```

Definition at line 85 of file acting.cpp.

```
{
    while(isRun())
    {
        Action* action = request_pool.pop();
        if(action == nullptr) continue;
        if(!isRun()) break;
        switch(action->isRequest())
        {
            case (REQUEST::ID::NONE):
                break;
            case (REQUEST::ID::INVALID):
                break;
            case (REQUEST::ID::CHANGE_INF):
                {
                    std::string id = action->getArgs().getInterfaceName();
                    UI.push(id);
                    break;
                }
            default:
                break;
        };

        delete action;
    }
}
```

7.24.3.11 run()

```
void Window::run ( )
```

Definition at line 4 of file running.cpp.

```
                 {
    // last_chrismas = now()
    Wcontent.setInputClock2Now();
    Wcontent.setRuntimeClock2Now();

    //Wcontent.thread_pool.push_back(std::thread(&Window::draw, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::getUserEvent, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::getRuntimeEvent, this));
    //Wcontent.thread_pool.push_back(std::thread(&Window::sound_effect, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::userActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::userActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::immediateActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::immediateActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::immediateActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::requestActing, this));

    while (isRun())
    {
        draw();
        systemEvent();
        systemActing();
        //getUserEvent();
        //getRuntimeEvent();
        //sound_effect();
        //userActing();
        //immediateActing();
        std::this_thread::sleep_for(std::chrono::milliseconds(5));
    }
}
```

7.24.3.12 sound_effect()

`void Window::sound_effect ( )  [protected]`

Definition at line 126 of file running.cpp.

```
{
    // do nothing
}
```

7.24.3.13 systemActing()

`void Window::systemActing ( )  [protected]`

Definition at line 74 of file acting.cpp.

```
{
    {
        Action* action = system_pool.pop();
        if(action == nullptr) return;
        if(!isRun()) return;
        action->execute();
        delete action;
    }
}
```

7.24.3.14 systemEvent()

`void Window::systemEvent ( )  [protected]`

Definition at line 44 of file running.cpp.

```
{
    {
        // alt + F4 to exit
        if (IsKeyDown(KEY_LEFT_ALT) && IsKeyDown(KEY_F4))
        {
            system_pool.push(new CloseAction(this));
        }
        if (WindowShouldClose())
        {
            system_pool.push(new CloseAction(this));
        }

        if (IsWindowResized() && !IsWindowFullscreen())
        {
            int width = GetScreenWidth();
            int height = GetScreenHeight();
            system_pool.push(new resizeAction(this, width, height));
        }
    }
}
```

7.24.3.15 userActing()

```
void Window::userActing ( ) [protected]
```

Definition at line 62 of file acting.cpp.

```
{
    while(isRun())
    {
        Action* action = immediate_user_pool.pop();
        if(action == nullptr) continue;
        if(!isRun()) break;
        action->execute();
        delete action;
    }
}
```

7.24.4 Friends And Related Symbol Documentation

7.24.4.1 CloseAction

```
friend class CloseAction [friend]
```

Definition at line 144 of file window.hpp.

7.24.4.2 resizeAction

```
friend class resizeAction [friend]
```

Definition at line 145 of file window.hpp.

The documentation for this class was generated from the following files:

- src/window/include/window.hpp
- src/window/src/acting.cpp
- src/window/src/constructor.cpp
- src/window/src/destructor.cpp
- src/window/src/running.cpp

Chapter 8

File Documentation

8.1 README.md File Reference

8.2 src/action/include/action.hpp File Reference

```
#include <vector>
#include <string>
```

Classes

- struct ARGS

  stores request information
- class Action

  manages the way an action is executed
- class PacketAction

  organize selected actions into a package

Variables

- ARGS NONE_ARGS

8.2.1 Variable Documentation

8.2.1.1 NONE_ARGS

```
ARGS NONE_ARGS  [extern]
```

Definition at line 4 of file action.cpp.

8.3 action.hpp

Go to the documentation of this file.

```
#ifndef ACTION_HPP
#define ACTION_HPP

#include <vector>
#include <string>

struct ARGS
{
    std::vector<std::string> str;
    std::vector<int> num;
    std::vector<void*> addr;
    ARGS() = default;
    ~ARGS() = default;

    std::string getInterfaceName();
};
extern ARGS NONE_ARGS;

class Action
{
public:
    Action();
    Action(Action*);
    virtual ~Action() = default;

    virtual int isRequest();
    virtual bool isPackage();
    virtual void execute();
    virtual Action* clone();
    virtual std::vector<Action*> unpack();
    virtual ARGS& getArgs();
};

class PacketAction : public Action
{
private:
    std::vector<Action*> actions;
public:
    PacketAction();
    PacketAction(PacketAction*);
    ~PacketAction();
    bool isPackage() override;
    void addAction(Action*);
    void addAction(PacketAction*);
    std::vector<Action*> unpack() override;
    void execute() override;
    PacketAction* clone() override;
};

#endif
```

8.4 src/action/include/request.hpp File Reference

```
#include <action.hpp>
```

Classes

- class Request

  sends information to a higher, relevant entity
- class changeInfRequest
- class loseRequest

  request sent when the player loses

8.5 request.hpp

Go to the documentation of this file.

```
#ifndef REQUEST_MY_HPP
#define REQUEST_MY_HPP

#include <action.hpp>

class Request : public Action
{
protected:
    ARGS args;
public:
    Request();
    Request(Request*);
    ~Request() = default;

    int isRequest() override;
    virtual Action* clone() override;
};

class changeInfRequest : public Request
{
public:
    changeInfRequest(std::string s);
    changeInfRequest(changeInfRequest*);
    ~changeInfRequest() = default;
    int isRequest() override;
    Action* clone() override;
    ARGS& getArgs() override;
};

class loseRequest : public Request
{
public:
    loseRequest() = default;
    loseRequest(loseRequest*);
    ~loseRequest() = default;
    int isRequest() override;
    Action* clone() override;
};

#endif
```

8.6 src/utils/include/const/request.hpp File Reference

Namespaces

- namespace REQUEST

Enumerations

- enum REQUEST::ID {
  REQUEST::INVALID , REQUEST::NONE , REQUEST::CHANGE_INF , REQUEST::DELAY ,
  REQUEST::LOSE }

8.7 request.hpp

Go to the documentation of this file.

```
#ifndef REQUEST_HPP
#define REQUEST_HPP

namespace REQUEST
{
    enum ID
    {
        INVALID,
        NONE,
        CHANGE_INF,
        DELAY,
        LOSE,
    };
}

#endif

```

8.8 src/action/src/action.cpp File Reference

```
#include <action.hpp>
#include <queue>
```

Variables

- ARGS NONE_ARGS

8.8.1 Variable Documentation

8.8.1.1 NONE_ARGS

```
ARGS NONE_ARGS
```

Definition at line 4 of file action.cpp.

8.9 action.cpp

Go to the documentation of this file.

```
00001 #include <action.hpp>
00002 #include <queue>
00003
00004 ARGS NONE_ARGS;
00005
00006 Action::Action()
00007 {
00008 }
00009
00010 Action::Action(Action* action)
00011 {
00012 }
00013
00014
00015 int Action::isRequest()
00016 {
00017     return 0;
00018 }
00019
00020 bool Action::isPackage()
00021 {
00022     return false;
00023 }
00024
00025 void Action::execute()
00026 {
00027 }
00028
00029 Action* Action::clone()
00030 {
00031     return this;
00032 }
```

```

std::vector<Action*> Action::unpack()
{
    return std::vector<Action*> ({this});
}

ARGS& Action::getArgs()
{
    return NONE_ARGS;
}

PacketAction::PacketAction() : Action()
{
}


PacketAction::PacketAction(PacketAction* action) : Action(action)
{
    for(Action* a : action->actions)
        actions.push_back(a->clone());
}

PacketAction::~PacketAction()
{
    for(Action* a : actions)
        delete a;
    actions.clear();
}

bool PacketAction::isPackage()
{
    return true;
}

void PacketAction::addAction(Action* action)
{
    actions.push_back(action);
}

void PacketAction::addAction(PacketAction* action)
{
    for(auto i : action->actions)
        actions.push_back(i);
    action->actions.clear();
}

std::vector<Action*> PacketAction::unpack()
{
    std::vector<Action*> unpacked;
    std::queue<PacketAction*> q;

    q.push(this);

    while(!q.empty())
    {
        PacketAction* p = q.front();
        q.pop();

        for(Action* a : p->actions)
        {
            if(a->isPackage())
            {
                q.push((PacketAction*)a);
            }
            else
            {
                unpacked.push_back(a);
            }
        }
        p->actions.clear();
    }
    return unpacked;
}

void PacketAction::execute()
{
    for(Action* a : actions)
    {
        a->execute();
    }
}


PacketAction* PacketAction::clone()
{
    return new PacketAction(this);
}
```

8.10 src/chunk/src/action.cpp File Reference

```
#include "action.hpp"
#include <chunk.hpp>
```

8.11 action.cpp

Go to the documentation of this file.

```
#include "action.hpp"
#include <chunk.hpp>

Container* Chunk::randomEntity()
{
    int value = GetRandomValue(0, 100);

    for(auto i : visiter)
    {
        value -= i->getProbability();
        if(value <= 0) return i;
    }

    return visiter[GetRandomValue(0, visiter.size() - 1)];
}

void Chunk::movingEntity()
{
    for(auto i : Entity)
    {
        i->moveBy(velocity);
        i->nextImage();
    }
}

Action* Chunk::getRuntimeEvent()
{
    PacketAction* packet = nullptr;
    Action* action = Interface::getRuntimeEvent();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    if(std::chrono::system_clock::now() - moveClock >= moveTime)
    {
        Action* action = new moveEntityAction(this);
        if(packet == nullptr)
        {
            packet = new PacketAction();
        }
        packet->addAction(action);
        moveClock = std::chrono::system_clock::now();
    }
    return packet;
}
```

8.12 src/game/src/action.cpp File Reference

```
#include <game.hpp>
```

8.13 action.cpp

Go to the documentation of this file.

```
#include <game.hpp>

Action* Game::react()
{
    return Interface::react();
}

Action* Game::getRuntimeEvent()
{
    // if now - mapSpeedClock < 10 millisecond, return nullptr

    if(std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::system_clock::now() -
mapSpeedClock).count() < 20)
        return nullptr;
    Action* action;
    PacketAction* packet = nullptr;
    action = Interface::getRuntimeEvent();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    action = new moveChunksAction(this, mapDisplacement);
    if(packet == nullptr) packet = new PacketAction();
    packet->addAction(action);

    for(auto i : chunks)
    {
        Action* act = i->getRuntimeEvent();
        if(act == nullptr)
            continue;
        if(packet == nullptr)
            packet = new PacketAction();
        packet->addAction(act);
    }

    action = new moveObjectAction(main, mapDisplacement);
    if(packet == nullptr) packet = new PacketAction();
    packet->addAction(action);

    mapSpeedClock = std::chrono::system_clock::now();

    return packet;
}
```

8.14 src/interface/src/action.cpp File Reference

```
#include "action.hpp"
#include <interface.hpp>
```

8.15 action.cpp

Go to the documentation of this file.

```
#include "action.hpp"
#include <interface.hpp>

Action* Interface::getRuntimeEvent()
{
    PacketAction* packet = nullptr;
    Action* action = Container::getRuntimeEvent();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    for(auto i : nested)
    {
        action = i->getRuntimeEvent();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    for(auto i : containers)
    {
        action = i->getRuntimeEvent();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    return packet;
}

Action* Interface::react()
{
    if(!isVisible()) return nullptr;
    PacketAction* packet = nullptr;

    Action* action = Container::react();

    if(action != nullptr)
    {
        packet = new PacketAction();
        packet->addAction(action);
    }

    for(auto i : keystrokes)
    {
        Action* action = i->react();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    for(auto i : containers)
    {
        Action* action = i->react();
        if(action != nullptr)
        {
            if(packet == nullptr) packet = new PacketAction();
            packet->addAction(action);
        }
    }

    return packet;
}
```

8.16 src/action/src/args.cpp File Reference

```
#include <action.hpp>
```

8.17 args.cpp

Go to the documentation of this file.

```
#include <action.hpp>

std::string ARGS::getInterfaceName()
{
    return str[0];
}
```

8.18 src/action/src/request.cpp File Reference

```
#include <request.hpp>
#include <const/request.hpp>
```

8.19 request.cpp

Go to the documentation of this file.

```
#include <request.hpp>
#include <const/request.hpp>

Request::Request() : Action()
{
}

Request::Request(Request* request) : Action(request)
{
}

int Request::isRequest()
{
    return 1;
}

Action* Request::clone()
{
    return new Request(this);
}
```

8.20 src/utils/src/request.cpp File Reference

8.21 request.cpp

Go to the documentation of this file.

8.22 src/action/src/request/changeinf.cpp File Reference

```
#include <request.hpp>
#include <const/request.hpp>
```

8.23 changeinf.cpp

Go to the documentation of this file.

```
#include <request.hpp>
#include <const/request.hpp>
changeInfRequest::changeInfRequest(std::string s)
{
    args.str.push_back(s);
}

changeInfRequest::changeInfRequest(changeInfRequest* other)
{
    args = other->args;
}
```

```

int changeInfRequest::isRequest()
{
    return REQUEST::CHANGE_INF;
}

Action* changeInfRequest::clone()
{
    return new changeInfRequest(this);
}

ARGS& changeInfRequest::getArgs()
{
    return args;
}
```

8.24 src/action/src/request/lose.cpp File Reference

```
#include <request.hpp>
#include <const/request.hpp>
```

8.25 lose.cpp

Go to the documentation of this file.

```
#include <request.hpp>
#include <const/request.hpp>

loseRequest::loseRequest(loseRequest* other)
{
    args = other->args;
}

int loseRequest::isRequest()
{
    return REQUEST::LOSE;
}

Action* loseRequest::clone()
{
    return new loseRequest(this);
}

```

8.26 src/button/include/button.hpp File Reference

```
#include <raylib.h>
#include <frame.hpp>
#include <container.hpp>
```

Classes

- class ButtonImage

    manages the appearance and behavior of a button

Macros

- #define TRANSPARENT Color {127, 127, 127, 0}
- #define rectangle this->getFrame()

8.26.1 Macro Definition Documentation

8.26.1.1 rectangle

```
#define rectangle this->getFrame()
```

Definition at line 10 of file button.hpp.

8.26.1.2 TRANSPARENT

```
#define TRANSPARENT Color {127, 127, 127, 0}
```

Definition at line 9 of file button.hpp.

8.27 button.hpp

Go to the documentation of this file.

```
00001 #ifndef BUTTON_HPP
00002 #define BUTTON_HPP
00003 
00004 #include <raylib.h>
00005 
00006 #include <frame.hpp>
00007 #include <container.hpp>
00008 
00009 #define TRANSPARENT Color {127, 127, 127, 0}
00010 #define rectangle this->getFrame()
00011 
00018 class ButtonImage : public Container
00019 {
00020 private:
00021     static constexpr int DPI = 500;
00022     static constexpr float CORNER_RADIUS = 0.3;
00023 
00024     std::vector <std::string> path;
00025     std::vector <std::string> pathPress;
00026 
00027     int numpath;
00028     int tmpPath;
00029     int releaseID;
00030     int hoverID;
00031     int pressingID;
00032     int clickedID;
00033 
00034     // Rectangle rectangle;
00035     Color color;
00036 
00037     bool isHover = false;
00038     bool pressing = false, clicked = false;
00039 
00040     std::vector <Action*> actions;
00041 
00042 protected:
00043     void loadEvent(YAML::Node node);
00044 
00045 
00046 public:
00047     ButtonImage(Frame* parrent, Rectangle relative);
00048     ~ButtonImage();
00049     void draw();
00050     PacketAction* react();
00051 
00052     void changeIndex(int newindex);
00053     void changePosition(Rectangle change);
00054     [[nodiscard]] bool isClicked() const;
00055     [[nodiscard]] bool isPressing() const;
00056     int getClicked();
00057 
00058     std::string linkContent(std::string);
00059     std::string linkContentAbsolute(std::string);
00060 
00061 };
00062 
00063 #endif
```

8.28 src/button/src/arthmetic.cpp File Reference

```
#include <button.hpp>
#include <request.hpp>
```

8.29 arthmetic.cpp

Go to the documentation of this file.

```
#include <button.hpp>
#include <request.hpp>


// Button for image

void ButtonImage::draw() {
    if(!isVisible()) return;
    this->Container::draw();
}

PacketAction* ButtonImage::react() {

    if (CheckCollisionPointRec(GetMousePosition(), rectangle)) {
        this->isHover = 1;
        if (IsMouseButtonDown(MOUSE_LEFT_BUTTON)) { // click -> pressing
            this->clicked = true;
            if(this->pressingID == -1)
                return nullptr;
            PacketAction* packet = new PacketAction();
            packet->addAction(actions[pressingID]->clone());
            return packet;
        }
        else if(this->clicked) { // release -> click

            this->clicked = false;
            if(this->clickedID == -1)
                return nullptr;
            PacketAction* packet = new PacketAction();
            packet->addAction(actions[clickedID]->clone());
            packet->addAction(new changeInfRequest("test"));
            return packet;
        }
        if(this->hoverID == -1)
            return nullptr;
        PacketAction* packet = new PacketAction();
            packet->addAction(actions[hoverID]->clone());
            return packet;
    }
    if (this->isHover == 1)
    {
        this->isHover = 0;
        if(this->releaseID == -1)
            return nullptr;
        PacketAction* packet = new PacketAction();
            packet->addAction(actions[releaseID]->clone());
            return packet;
    }
    return nullptr;
}



void ButtonImage::changeIndex(int newindex)
{
    tmpPath = newindex;
}

int ButtonImage::getClicked()
{
    return tmpPath;
}

bool ButtonImage::isClicked() const {
    return this->clicked;
}
```

8.30 src/chunk/src/arthmetic.cpp File Reference

```
#include <chunk.hpp>
```

8.31 arthmetic.cpp

Go to the documentation of this file.

```
#include <chunk.hpp>

void Chunk::drawEntity()
{
    for(auto i : Entity)
        i->draw();
}

void Chunk::draw()
{

    Container::draw();
    drawNested();
    drawContainers();
    drawEntity();
}
```

8.32 src/container/src/arthmetic.cpp File Reference

```
#include "action.hpp"
#include <container.hpp>
```

8.33 arthmetic.cpp

Go to the documentation of this file.

```
#include "action.hpp"
#include <container.hpp>

void Container::draw()
{
    if(sprites.empty()) return;
    if(!visible) return;
    sprites[focus[0]][focus[1]]->draw();
}

void Container::show()
{
    visible = true;
}

void Container::hide()
{
    visible = false;
}

void Container::toggleVisibility()
{
    visible = !visible;
}

bool Container::isVisible()
{
    return visible;
}

int Container::getInstanceId()
```

```
{
    return instance_id;
}

Action* Container::react()
{
    return nullptr;
}

Action* Container::getRuntimeEvent()
{
    return nullptr;
}
```

8.34 src/frame/src/arthmetic.cpp File Reference

```
#include <frame.hpp>
```

8.35 arthmetic.cpp

Go to the documentation of this file.

```
#include <frame.hpp>

void Frame::updateFrame(bool recursive)
{

    if(parent != nullptr)
    {
        std::lock_guard<std::mutex> lock(mtx);
        frame.x = parent->getX() + relative[0] * parent->getW();
        frame.y = parent->getY() + relative[1] * parent->getH();
        frame.width = relative[2] * parent->getW();
        frame.height = relative[3] * parent->getH();
    }

    if(recursive)
    {
        for(auto& subframe : subframes)
        {
            subframe->updateFrame(true);
        }
    }
}

void Frame::moveTo(fPoint rel)
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] = rel[0];
    relative[1] = rel[1];
    mtx.unlock();
    updateFrame(true);
}
void Frame::moveTo(int x, int y)
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x = x;
    frame.y = y;
    mtx.unlock();
    updateFrame(true);
}

void Frame::moveCenterTo(fPoint rel)
{
    if(isroot()) return ;
    mtx.lock();
    fPoint center = getCenter();
    relative[0] += rel[0] - center[0];
    relative[1] += rel[1] - center[1];
    mtx.unlock();
    updateFrame(true);
}

```

```
void Frame::moveCenterTo(int x, int y)
{
    if(parent != nullptr) return ;
    mtx.lock();
    fPoint center = getCenter();
    frame.x += x - center[0];
    frame.y += y - center[1];
    mtx.unlock();
    updateFrame(true);
}

void Frame::moveBy(fPoint rel)
{
    if(isroot()) return ;
    mtx.lock();
    relative[0] += rel[0];
    relative[1] += rel[1];
    mtx.unlock();
    updateFrame(true);
}

void Frame::moveBy(int x, int y)
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.x += x;
    frame.y += y;
    mtx.unlock();
    updateFrame(true);
}

void Frame::resize(fPoint rel)
{
    if(isroot()) return ;
    mtx.lock();
    relative[2] = rel[0];
    relative[3] = rel[1];
    mtx.unlock();
    updateFrame(true);
}

void Frame::resize(int w, int h)
{
    if(parent != nullptr) return ;
    mtx.lock();
    frame.width = w;
    frame.height = h;
    mtx.unlock();
    updateFrame(true);
}

const Rectangle& Frame::getFrame() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}

const fRect& Frame::getRelative() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}

Frame* Frame::getParent()
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent;
}

void Frame::setRelative(fRect rel)
{
    mtx.lock();
    relative = rel;
    mtx.unlock();
    updateFrame(true);
}

const fPoint& Frame::getCenter() const
{
    std::lock_guard<std::mutex> lock(mtx);
    static fPoint resu;
    if(isroot())
        resu = {frame.x + frame.width / 2, frame.y + frame.height / 2};
    else
        resu = {relative[0] + relative[2] / 2, relative[1] + relative[3] / 2};

    return resu;
}

const float& Frame::getX() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.x;
}

const float& Frame::getY() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.y;
}

const float& Frame::getW() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.width;
}

const float& Frame::getH() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame.height;
}

Frame::operator Rectangle() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return frame;
}

Frame::operator fRect() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return relative;
}

Frame::operator iRect() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return {(int) frame.x, (int) frame.y, (int) frame.width, (int) frame.height};
}


```

8.36 src/game/src/arthmetic.cpp File Reference

```
#include <game.hpp>
```

8.37 arthmetic.cpp

Go to the documentation of this file.

```
#include <game.hpp>

void Game::draw()
{

    drawNested();

    for(auto i = chunks.begin(); i != chunks.end(); ++i)
    {
        (*i)->draw();
    }

    drawContainers();
}
```

8.38 src/interface/src/arthmetic.cpp File Reference

```
#include <interface.hpp>
```

8.39 arthmetic.cpp

Go to the documentation of this file.

```
#include <interface.hpp>

void Interface::drawNested()
{
    for(auto& child : nested)
    {
        child->draw();
    }
}

void Interface::drawContainers()
{
    for(auto& child : containers)
    {
        child->draw();
    }
}

void Interface::draw()
{
    Container::draw();

    drawNested();

    drawContainers();


}
```

8.40 src/object/src/arthmetic.cpp File Reference

```
#include "container.hpp"
#include <object.hpp>
```

8.41 arthmetic.cpp

Go to the documentation of this file.

```
#include "container.hpp"
#include <object.hpp>
Action* Object::react()
{
    if(std::chrono::steady_clock::now() < waitUntil)
        return nullptr;
    for(int i = 0; i < strokes.size(); i++)
    {
        Action* a = strokes[i].stroke->react();
        if(a == nullptr) continue;
        else strokes[i].stroke->nextAction();
        return a;
    }


    return nullptr;
}

void Object::draw()
{
    Container::draw();
    return ;
}
```

8.42 src/visual/src/arthmetic.cpp File Reference

```
#include <visual.hpp>
```

8.43 arthmetic.cpp

Go to the documentation of this file.

```
#include <visual.hpp>


void Visual::draw()
{
    if(m_texture == nullptr) return ;
    Rectangle rec = getFrame();
    // draw texture
    DrawTexture(*m_texture, rec.x, rec.y, WHITE);
}

void Visual::fitFrame()
{
    if(m_texture == nullptr) return ;
    if(!resizeable) return ;
    const Rectangle &rec = Frame::getFrame();

    Image img = LoadImageFromTexture(*m_texture);
    UnloadTexture(*m_texture.get());

    ImageResize(&img, rec.width, rec.height);
    *m_texture.get() = LoadTextureFromImage(img);
    UnloadImage(img);
}

void Visual::resize(fPoint rel)
{
    Frame::resize(rel);
    updateFrame(true);
}

void Visual::updateFrame(bool recursive)
{
    if(m_texture == nullptr) return ;
    float prx = getFrame().width;
    float pry = getFrame().height;
    Frame::updateFrame(recursive);

    float rx = getFrame().width - prx;
    float ry = getFrame().height - pry;
    if(rx < 1e-3 && ry < 1e-3) return ;
    fitFrame();
}
```

8.44 src/button/src/constructor.cpp File Reference

```
#include <button.hpp>
```

8.45 constructor.cpp

Go to the documentation of this file.

```
#include <button.hpp>


// Button for Image
ButtonImage::ButtonImage(Frame* parrent, Rectangle rel) : Container(parrent, rel)
{
     // set default
    this->chooseImage(0, this->tmpPath);
    this->color = WHITE;
    this->pressing = false;
    this->isHover = false;
    this->clicked = false;

    this->releaseID = -1;
    this->hoverID = -1;
    this->pressingID = -1;
    this->clickedID = -1;
```

```cpp
}

std::string ButtonImage::linkContent(std::string path)
{
    return linkContentAbsolute(PATB::BUTTON_ + path);
}

std::string ButtonImage::linkContentAbsolute(std::string path)
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
    {
        loadSprites(node["textures"]);
        chooseImage(0, 0);
    }
    if(node["events"])
    {
        loadEvent(node["events"]);
    }

    return getName();
}

void ButtonImage::loadEvent(YAML::Node node)
{
    if(node["hover"])
    {

        for(auto sprite : node["hover"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));
        }
        this->hoverID = actions.size() - 1;
    }

    if(node["release"])
    {
        for(auto sprite : node["release"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));

        }
        this->releaseID = actions.size() - 1;
    }

    if(node["clicked"])
    {
        for(auto sprite : node["clicked"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));
        }
        this->clickedID = actions.size() - 1;
    }

    if(node["pressing"])
    {
        for(auto sprite : node["pressing"]["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            actions.push_back(new changeImageAction(this, p));
        }
        this->pressingID = actions.size() - 1;
    }
}
```

8.46 src/chunk/src/constructor.cpp File Reference

```
#include "const/path/atb.hpp"
#include <chunk.hpp>
```

8.47 constructor.cpp

Go to the documentation of this file.

```
#include "const/path/atb.hpp"
#include <chunk.hpp>

Chunk::Chunk(Frame* frame, Rectangle rect) : Interface(frame, rect)
{

}

Chunk::Chunk(Chunk* other) : Interface(other)
{

    for(auto i : other->visiter)
    {
        Rectangle rel;
        rel.x = 1;
        rel.y = -0.375;
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        visiter.push_back(new Container(i, this, rel));
    }
    velocity = other->velocity;
    generateEntity();
}

Chunk::Chunk(Chunk* other, Rectangle rect) : Interface(other, rect)
{
    for(auto i : other->visiter)
    {
        Rectangle rel;
        rel.x = 1;
        rel.y = -0.375;
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        visiter.push_back(new Container(i, this, rel));
    }
    velocity = other->velocity;
    generateEntity();
}

Chunk::Chunk(Chunk* other, Frame* frame, Rectangle rect) : Interface(other, frame, rect)
{
    for(auto i : other->visiter)
    {
        Rectangle rel;
        rel.x = 1;
        rel.y = -0.375;
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        visiter.push_back(new Container(i, this, rel));
    }
    velocity = other->velocity;
    generateEntity();
}

void Chunk::generateEntity()
{

    if(visiter.empty()) return;
    float x = GetRandomValue(-40, 10);

    while(x < 0.9)
    {
        Container* c = randomEntity();
        Rectangle rel;
        rel.x = x;
        rel.y = -0.375;
        rel.width = c->getRelative()[2];
        rel.height = c->getRelative()[3];
        Container* cont = new Container(c, this, rel);
        Entity.push_back(cont);
        x += GetRandomValue(20, 60) / 100.0;
    }
}


std::string Chunk::linkContent(std::string path)
{
    return linkContentAbsolute(PATB::CHUNK_ + path);
}


void Chunk::addVisiter(Container* obj)
{
    Rectangle rel;
    rel.x = obj->getRelative()[0];
    rel.y = obj->getRelative()[1];
    rel.width = obj->getRelative()[2];
    rel.height = obj->getRelative()[3];

    Container* c = new Container(obj, this, rel);
    visiter.push_back(c);
}

void Chunk::addVisiter(Container* obj, int prob)
{
    Rectangle rel;
    rel.x = obj->getRelative()[0];
    rel.y = obj->getRelative()[1];
    rel.width = obj->getRelative()[2];
    rel.height = obj->getRelative()[3];

    Container* c = new Container(obj, this, rel);
    c->setProbability(prob);
    visiter.push_back(c);
}

void Chunk::addVisiter(Container* obj, Rectangle rel)
{
    Container* c = new Container(obj, this, rel);
    visiter.push_back(c);
}

void Chunk::addVisiter(Container* obj, int prob, Rectangle rel)
{
    Container* c = new Container(obj, this, rel);
    c->setProbability(prob);
    visiter.push_back(c);
}

void Chunk::setVelocity(fPoint vel)
{
    velocity = vel;
}
```

8.48 src/container/src/constructor.cpp File Reference

```
#include "raylib.h"
#include <visual.hpp>
#include <container.hpp>
#include <const/path/atb.hpp>
#include <const/path/assets.hpp>
#include <file.hpp>
```

8.49 constructor.cpp

Go to the documentation of this file.

```
#include "raylib.h"
#include <visual.hpp>
#include <container.hpp>
#include <const/path/atb.hpp>
#include <const/path/assets.hpp>
#include <file.hpp>

int Container::id_count = 0;

Container::Container(Frame* parent, Rectangle rect) : Frame(parent, rect)
{
    instance_id = id_count++;
    focus = {0, 0};
    visible = true;
}

Container::Container(Container* other) : Frame(other)
{
    instance_id = id_count++;
    focus = {0, 0};
    name = other->name;
    visible = true;

    for(auto s : other->sprites)
    {
        sprites.emplace_back();
        Rectangle rect;
        rect.x = other->getRelative()[0];
        rect.y = other->getRelative()[1];
        rect.width = other->getRelative()[2];
        rect.height = other->getRelative()[3];

        for(auto v : s)
        {
            sprites.back().push_back(new Visual(v, this, rect));
        }
    }
}

Container::Container(Container* other, Rectangle rect) : Frame(other)
{
    instance_id = id_count++;
    focus = {0, 0};
    name = other->name;
    setRelative({rect.x, rect.y, rect.width, rect.height});
    visible = true;

    for(auto s : other->sprites)
    {
        sprites.emplace_back();
        Rectangle rect;
        rect.x = other->getRelative()[0];
        rect.y = other->getRelative()[1];
        rect.width = other->getRelative()[2];
        rect.height = other->getRelative()[3];

        for(auto v : s)
        {
            sprites.back().push_back(new Visual(v, this, rect));
        }
    }
}

Container::Container(Container* other, Frame* parent, Rectangle rect) : Frame(parent, rect)
{
    instance_id = id_count++;
    focus = {0, 0};
    name = other->name;
    visible = true;
    for(auto s : other->sprites)
    {
        sprites.emplace_back();
        Rectangle rect;
        rect.x = other->getRelative()[0];
        rect.y = other->getRelative()[1];
        rect.width = other->getRelative()[2];
        rect.height = other->getRelative()[3];

        for(auto v : s)
        {
            sprites.back().push_back(new Visual(v, this, rect));
        }
    }
}

std::string Container::linkContent(std::string path)
{
    focus = {0, 0};
    return linkContentAbsolute(PATB::CONTAINER_ + path);
}

std::string Container::linkContentAbsolute(std::string path)
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
    {

        loadSprites(node["textures"]);
    }

    if(node["focus"])
    {
        loadFocus(node["focus"]);
    }

    return name;
}

bool Container::loadName(YAML::Node node)
{
    if(!node["name"])
    {
        name = "";
        return false;
    }
    name = node["name"].as<std::string>();
    return true;
}

void Container::loadSprites(YAML::Node node)
{
    for(auto sprite : node)
    {
        if(!sprite["path"]) continue;
        if(!sprite["graphics"]) continue;

        std::string path = PASSETS::GRAPHIC_ + sprite["path"].as<std::string>();
        Image image = LoadImage(path.c_str());

        if(sprite["resize"])
        {
            int x = image.width * sprite["resize"][0].as<float>();
            int y = image.height * sprite["resize"][1].as<float>();
            ImageResize(&image, x, y);
        }

        sprites.emplace_back();
        for(auto img : sprite["graphics"])
        {
            float x, y, w, h;
            int repeat = 1;
            int gapX = 0;
            int gapY = 0;

            int dx = 1;
            int dy = 1;

            if(img["x"])
                x = img["x"].as<float>() / 100.0;
            else x = 0;
            if(img["y"])
                y = img["y"].as<float>() / 100.0;
            else y = 0;
            if(img["w"])
                w = img["w"].as<float>() / 100.0;
            else w = 1;
            if(img["h"])
                h = img["h"].as<float>() / 100.0;
            else h = 1;
            if(img["repeat"])
                repeat = img["repeat"].as<int>();
            if(img["gapX"])
                gapX = img["gapX"].as<int>();
            if(img["gapY"])
                gapY = img["gapY"].as<int>();

            if(img["dx"])
                dx = img["dx"].as<int>();
            if(dx < 0) dx = -1;
            else dx = 1;

            if(img["dy"])
```

```
                    dy = img["dy"].as<int>();
                if(dy < 0) dy = -1;
                else dy = 1;

                int imgw = image.width;
                int imgh = image.height;

                if(img["axis"] && img["axis"].as<std::string>() == "horizontal")
                {
                    for(float j = y; j >= 0 && j + h < 1 + 1e-2; j += dy * (gapY + h))
                    {
                        for(float i = x; i >= 0 && i + w <= 1 + 1e-2 && repeat--; i += dx * (gapX + w))
                        {
                            Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                            Image img2 = ImageFromImage(image, rect);
                            Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                            Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                            sprites.back().push_back(vis);

                            UnloadImage(img2);
                        }
                    }
                }else
                {
                    for(float i = x; i >= 0 && i + w <= 1 + 1e-2; i += dx * (gapX + w))
                    {
                        for(float j = y; j >= 0 && j + h < 1 + 1e-2 && repeat--; j += dy * (gapY + h))
                        {
                            Rectangle rect = {i * imgw, j * imgh, w * imgw, h * imgh};
                            Image img2 = ImageFromImage(image, rect);
                            Texture2D *txt = new Texture2D(LoadTextureFromImage(img2));
                            Visual *vis = new Visual(txt, this, {0, 0, 1, 1});
                            sprites.back().push_back(vis);

                            UnloadImage(img2);
                        }
                    }
                }
            }
            UnloadImage(image);
        }
}

void Container::loadFocus(YAML::Node node)
{
    focus[0] = node[0].as<int>();
    focus[1] = node[1].as<int>();
}

void Container::chooseSprite(int index)
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[0] = index;
}

void Container::chooseImage(int index)
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    focus[1] = index;
}

void Container::chooseImage(int index, int index2)
{
    if(sprites.empty()) return;
    if(index < 0 || index >= sprites.size()) return;
    if(index2 < 0 || index2 >= sprites.at(index).size()) return;
    focus[0] = index;
    focus[1] = index2;
}

void Container::nextImage()
{
    if(sprites.empty()) return;
    focus[1]++;
    if(focus[1] >= sprites.at(focus[0]).size()) focus[1] = 0;
}

void Container::prevImage()
{
    if(sprites.empty()) return;
    focus[1]--;
    if(focus[1] < 0) focus[1] = sprites.at(focus[0]).size() - 1;
}

void Container::nextSprite()
```

```
{
    if(sprites.empty()) return;
    focus[0]++;
    if(focus[0] >= sprites.size()) focus[0] = 0;
}

void Container::prevSprite()
{
    if(sprites.empty()) return;
    focus[0]--;
    if(focus[0] < 0) focus[0] = sprites.size() - 1;
}

std::string Container::getName()
{
    return name;
}

void Container::setProbability(int prob)
{
    probability = prob;
}

int Container::getProbability()
{
    return probability;
}
```

8.50 src/frame/src/constructor.cpp File Reference

```
#include <frame.hpp>
```

8.51 constructor.cpp

Go to the documentation of this file.

```
#include <frame.hpp>

Frame::Frame(Frame* par, Rectangle rel)
{
    parent = nullptr;
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::Frame(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    parent = par;
    relative[0] = rel.x;
    relative[1] = rel.y;
    relative[2] = rel.width;
    relative[3] = rel.height;

    parent->addSubframe(this);

    updateFrame();
}
Frame::Frame(Frame* self)
{
    parent = nullptr;
    if(self == nullptr)
    {
        throw std::runtime_error("Frame::Frame(Frame* self): self is nullptr");
        return ;
    }
    parent = self->parent;
    relative = self->relative;
    frame = self->frame;
    for(auto& i : self->subframes)
    {
        subframes.push_back(i);
    }
}
Frame::Frame(Rectangle rec)
{
    parent = nullptr;
    frame = rec;
    parent = nullptr;
    relative = {1, 1, 1, 1};
}

```

8.52 src/game/src/constructor.cpp File Reference

```
#include "raylib.h"
#include <const/path/atb.hpp>
#include <file.hpp>
#include <vector.hpp>
#include <object.hpp>
#include <chunk.hpp>
#include <game.hpp>
```

8.53 constructor.cpp

Go to the documentation of this file.

```
#include "raylib.h"
#include <const/path/atb.hpp>
#include <file.hpp>
#include <vector.hpp>
#include <object.hpp>
#include <chunk.hpp>
#include <game.hpp>

Game::Game(Frame* frame, Rectangle rect) : Interface(frame, rect)
{
    initState = true;
}

Game::Game(Game* other) : Interface(other)
{
    initState = true;
}

Game::Game(Game* other, Rectangle rect) : Interface(other, rect)
{
    initState = true;
}

Game::Game(Game* other, Frame* frame, Rectangle rect) : Interface(other, frame, rect)
{
    initState = true;
}

std::string Game::linkContentAbsolute(std::string path)
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
        loadSprites(node["textures"]);

    if(node["focus"])
        loadFocus(node["focus"]);
    else chooseImage(0, 0);

    if(node["object"])
    {
        loadObject(node["object"]);
        for(int i = 0; i < getContainersSize(); i++)
            getContainers(i)->hide();
        main = getContainers(0);
        main->show();
    }
    if(node["collide"])
        loadCollide(node["collide"]);

    if(node["chunk"])
```

```
        loadChunk(node["chunk"]);

    if(node["attach-object"])
        loadAttactObject(node["attach-object"]);

    if(node["control"])
        loadControl(node["control"]);

    if(node["event"])
        loadEvent(node["event"]);

    if(node["button"])
        loadButton(node["button"]);
    return getName();
}

void Game::loadCollide(YAML::Node node)
{
}

void Game::loadMap()
{
    if(cache.empty()) return ;
    while(!chunks.empty())
    {
        fRect rec = chunks.back()->getRelative();
        if(rec[1] > 1) chunks.pop_back();
        else break;
    }
    if(chunks.empty())
    {
        Rectangle rel;
        rel.width = cache[0]->getRelative()[2];
        rel.height = cache[0]->getRelative()[3];
        rel.x = 0;
        rel.y = (1.01 - rel.height);

        Chunk* chunk = new Chunk(cache[0], this, rel);
        chunks.push_front(chunk);
        for(int i = 0; i < 3; i++)
        {
            rel.y += 0.005 - rel.height;
            chunk = new Chunk(cache[0], this, rel);
            chunks.push_front(chunk);
        }
    }
    while(chunks.front()->getRelative()[1] > 0)
    {
        Rectangle rel;
        rel.width = chunks.front()->getRelative()[2];
        rel.height = chunks.front()->getRelative()[3];
        rel.x = 0;
        rel.y = (chunks.front()->getRelative()[1] + 0.005 - rel.height);

        int id = GetRandomValue(0, cache.size() - 1);
        Chunk* chunk = new Chunk(cache[id], this, rel);
        chunks.push_front(chunk);
    }
}

void Game::loadChunk(YAML::Node node)
{
    for(auto i : node)
    {
        float x = 0, y = 0, w = 1, h = 1;
        int repeat = 1;
        std::string path = i["file"].as<std::string>();
        if(i["x"]) x = i["x"].as<float>() / 100;
        if(i["y"]) y = i["y"].as<float>() / 100;
        if(i["w"]) w = i["w"].as<float>() / 100;
        if(i["h"]) h = i["h"].as<float>() / 100;
        if(i["repeat"]) repeat = i["repeat"].as<int>();
        fPoint direction = {1, 0};
        float velo = 0.002;
        if(i["velocity"])
        {
            velo = i["velocity"][0].as<float>();
            direction = {i["velocity"][1].as<float>(), i["velocity"][2].as<float>()};
        }
        float angle = VECTOR2D::getAngle(direction);
        fPoint displacement = {velo * cos(angle), velo * sin(angle)};
        Chunk* chunk = new Chunk(this, {x, y, w, h});
        chunk->linkContent(path);
        chunk->setVelocity(displacement);
        cache.push_back(chunk);
        while(--repeat > 0)
            cache.push_back(new Chunk(cache[0]));
    }
}

void Game::loadAttractObject(YAML::Node node)
{
    for(auto i : node)
    {
        int id = i["chunk"].as<int>();
        int objID = i["object"][0].as<int>();
        int prob = i["object"][1].as<int>();
        Container* container = getContainers(objID);
        container->setProbability(prob);
        cache[id]->addVisiter(container);
    }
}

void Game::loadEvent(YAML::Node node)
{
    if(node["map-speed"])
    {
        mapSpeed = node["map-speed"].as<float>();
    }
    if(node["map-direction"])
    {
        mapDirection[0] = node["map-direction"][0].as<float>();
        mapDirection[1] = node["map-direction"][1].as<float>();
    }
    float angle = VECTOR2D::getAngle(mapDirection);
    std::cout « "hehe: " « angle « std::endl;
    mapDisplacement[0] = mapSpeed * cos(angle);
    mapDisplacement[1] = mapSpeed * sin(angle);
}
```

8.54 src/interface/src/constructor.cpp File Reference

```
#include "raylib.h"
#include <interface.hpp>
#include <const/path/atb.hpp>
#include <file.hpp>
#include <object.hpp>
#include <chunk.hpp>
```

8.55 constructor.cpp

Go to the documentation of this file.

```
#include "raylib.h"
#include <interface.hpp>
#include <const/path/atb.hpp>
#include <file.hpp>
#include <object.hpp>
#include <chunk.hpp>

Interface::Interface(Frame* frame, Rectangle rect) : Container(frame, rect)
{
}

Interface::Interface(Interface* other) : Container(other)
{
    for(auto i : other->nested)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        nested.push_back(new Interface(i, this, rel));
    }
    for(auto i : other->containers)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        containers.push_back(new Container(i, this, rel));
    }
}

Interface::Interface(Interface* other, Rectangle rect) : Container(other, rect)
{
    for(auto i : other->nested)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        nested.push_back(new Interface(i, this, rel));
    }
    for(auto i : other->containers)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        containers.push_back(new Container(i, this, rel));
    }
}

Interface::Interface(Interface* other, Frame* frame, Rectangle rect) : Container(other, frame, rect)
{
    for(auto i : other->nested)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        nested.push_back(new Interface(i, this, rel));
    }
    for(auto i : other->containers)
    {
        Rectangle rel;
        rel.x = i->getRelative()[0];
        rel.y = i->getRelative()[1];
        rel.width = i->getRelative()[2];
        rel.height = i->getRelative()[3];
        containers.push_back(new Container(i, this, rel));
    }
}

std::string Interface::linkContent(std::string path)
{
    return linkContentAbsolute(PATB::INTERFACE_ + path);
}

std::string Interface::linkContentAbsolute(std::string path)
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";

    if(node["textures"])
        loadSprites(node["textures"]);

    if(node["focus"])
        loadFocus(node["focus"]);
    else chooseImage(0, 0);

    if(node["object"])
        loadObject(node["object"]);

    if(node["control"])
        loadControl(node["control"]);

    if(node["button"])
        loadButton(node["button"]);

//    if(node["collide"])
//        loadCollide(node["collide"]);

//    if(node["chunk"])
//        loadChunk(node["chunk"]);


//    if(node["event"])
//        loadEvent(node["event"]);

```

```
    return getName();
}

void Interface::loadObject(YAML::Node node)
{
    for(auto i : node)
    {
        Rectangle rel({0, 0, 0, 0});
        if(i["x"]) rel.x = i["x"].as<float>() / 100;
        if(i["y"]) rel.y = i["y"].as<float>() / 100;
        if(i["w"]) rel.width = i["w"].as<float>() / 100;
        if(i["h"]) rel.height = i["h"].as<float>() / 100;
        Container *obj;
        obj = new Object(this, rel);
        obj->linkContent(i["path"].as<std::string>());
        containers.push_back(obj);
    }
}


void Interface::loadControl(YAML::Node node)
{
    for(auto stroke : node)
    {
        KeyStroke* k = new KeyStroke();
        for(auto key : stroke["key"])
        {
            k->add(toKey(key.as<std::string>()));
        }
        std::string action = stroke["action"].as<std::string>();

        if(action == "move-object")
        {
            int id = stroke["args"][0].as<int>();
            float v = stroke["args"][1].as<float>() / 100.0;
            float x = stroke["args"][2].as<float>();
            float y = stroke["args"][3].as<float>();
            moveObjectAction* action = new moveObjectAction(containers[id], fPoint({x, y}), v);
            k->addAction(action);
        }

        keystrokes.push_back(k);
    }
}

void Interface::loadButton(YAML::Node node)
{
    for(auto i : node)
    {
        Rectangle rel({0, 0, 0, 0});
        if(i["x"]) rel.x = i["x"].as<float>() / 100;
        if(i["y"]) rel.y = i["y"].as<float>() / 100;
        if(i["w"]) rel.width = i["w"].as<float>() / 100;
        if(i["h"]) rel.height = i["h"].as<float>() / 100;
        ButtonImage *obj;
        obj = new ButtonImage(this, rel);
        obj->linkContent(i["path"].as<std::string>());
        obj->show();
        containers.push_back(obj);
    }
}

Container* Interface::getContainers(int id)
{
    if(id < 0 || id >= containers.size()) return nullptr;
    return containers[id];
}

int Interface::getContainersSize()
{
    return containers.size();
}
```

8.56 src/object/src/constructor.cpp File Reference

```
#include "container.hpp"
#include <object.hpp>
#include <const/path/atb.hpp>
#include <file.hpp>
```

8.57 constructor.cpp

Go to the documentation of this file.

```
#include "container.hpp"
#include <object.hpp>
#include <const/path/atb.hpp>
#include <file.hpp>

Object::Object(Frame* f, Rectangle rel) : Container(f, rel)
{
    waitUntil = std::chrono::steady_clock::now();
}

Object::Object(Object* other) : Container(other)
{
    waitUntil = std::chrono::steady_clock::now();
}

Object::Object(Object* other, Rectangle rel) : Container(other, rel)
{
    waitUntil = std::chrono::steady_clock::now();
}

Object::Object(Object* other, Frame* f, Rectangle rel) : Container(other, f, rel)
{
    waitUntil = std::chrono::steady_clock::now();
}

std::string Object::linkContent(std::string path)
{
    return linkContentAbsolute(PATB::OBJECT_ + path);
}

std::string Object::linkContentAbsolute(std::string path)
{
    YAML::Node node = YAML_FILE::readFile(path);
    if(!loadName(node)) return "";
    if(node["textures"]) loadSprites(node["textures"]);
    if(node["control"]) loadControl(node["control"]);

    chooseImage(0, 0);
    if(node["focus"])
        loadFocus(node["focus"]);

    return "";
}

void Object::loadControl(YAML::Node node)
{
    for(auto stroke : node)
    {
        strokes.emplace_back();
        KeyStroke* k = new KeyStroke();
        for(auto key : stroke["key"])
        {
            k->add(toKey(key.as<std::string>()));
        }
        for(auto sprite : stroke["sprite"])
        {
            iPoint p;
            int delay = 0;
            p[0] = sprite[0].as<int>();
            p[1] = sprite[1].as<int>();
            if(p.size() >= 3)
                delay = sprite[2].as<int>();
            k->addAction(new changeImageAction(this, p));
        }
        strokes.back().stroke = k;
    }
}
```

8.58 src/visual/src/constructor.cpp File Reference

```
#include <visual.hpp>
```

8.59 constructor.cpp

Go to the documentation of this file.

```
#include <visual.hpp>


Visual::Visual(Texture2D* txtr, Frame* frame, Rectangle rect) : Frame(frame, rect)
{
    m_texture = std::shared_ptr<Texture2D>(txtr, [](Texture2D* texture){
        UnloadTexture(*texture);
        texture = nullptr;
    });
    resizeable = true;
    fitFrame();
}

Visual::Visual(Visual* visual) : Frame(visual)
{
    resizeable = false;
    m_texture = visual->m_texture;
    fitFrame();
}

Visual::Visual(Visual* visual, Rectangle rect) : Frame(visual, rect)
{
    resizeable = false;
    m_texture = visual->m_texture;
    fitFrame();
}

Visual::Visual(Visual* visual, Frame* frame, Rectangle rect) : Frame(frame, rect)
{
    resizeable = false;
    m_texture = visual->m_texture;
    fitFrame();
}
```

8.60 src/window/src/constructor.cpp File Reference

```
#include <window.hpp>
#include <game.hpp>
#include <const/path/atb.hpp>
#include <file.hpp>
```

8.61 constructor.cpp

Go to the documentation of this file.

```
#include <window.hpp>
#include <game.hpp>
#include <const/path/atb.hpp>

#include <file.hpp>

Window::Window()
{
    Wcontent.width = 1200;
    Wcontent.height = 668;
    Wcontent.title = "Crossy Road clone";
    SetConfigFlags(FLAG_WINDOW_RESIZABLE | FLAG_VSYNC_HINT);
    InitWindow(Wcontent.width, Wcontent.height, Wcontent.title.c_str());
    SetTargetFPS(60);

    UI.setRootFrame(new Frame({0, 0, Wcontent.width, Wcontent.height}));
}

Window::Window(std::string path)
{
    path = PATB::WINDOW_ + path;
    YAML::Node config = YAML_FILE::readFile(path);
    initRaylib(config);
```

```cpp

    loadInterface(config["interface-list"]);
    loadGame(config["game"]);
    if(config["choose-interface"])
        UI.push(config["choose-interface"].as<std::string>());

    if(config["input-delay"])
    {
        double delay = config["input-delay"].as<int>() / 1000.0;
        Wcontent.input_delay = std::chrono::duration<double>(delay);
    }else
    {
        Wcontent.input_delay = std::chrono::duration<int>(50) / 1000.0;
    }

    if(config["runtime-delay"])
    {
        double delay = config["runtime-delay"].as<int>() / 1000.0;
        Wcontent.runtime_delay = std::chrono::duration<double>(delay);
    }else
    {
        Wcontent.runtime_delay = std::chrono::duration<int>(40) / 1000.0;
    }
}


void Window::initRaylib(YAML::Node config)
{
    Wcontent.width = config["width"].as<int>();
    Wcontent.height = config["height"].as<int>();
    Wcontent.title = config["title"].as<std::string>();
    // enable resizeable window and vsync

    SetConfigFlags(FLAG_WINDOW_RESIZABLE | FLAG_VSYNC_HINT);
    InitWindow(Wcontent.width, Wcontent.height, Wcontent.title.c_str());
    SetTargetFPS(60);
    Wcontent.setStatus(true);
    UI.setRootFrame(new Frame({0, 0, Wcontent.width, Wcontent.height}));
}

void Window::loadInterface(YAML::Node node)
{
    if(!node) return ;

    UI.setInterfacePool(new InterfacePool());
    for(auto i : node)
    {
        std::string path = i["file"].as<std::string>();
        float x = 0, y = 0, w = 1, h = 1;

        if(i["x"]) x = i["x"].as<float>() / 100;
        if(i["y"]) y = i["y"].as<float>() / 100;
        if(i["w"]) w = i["w"].as<float>() / 100;
        if(i["h"]) h = i["h"].as<float>() / 100;


        Interface* inf = new Interface(UI.getRootFrame(), {x, y, w, h});
        inf->linkContent(path);
        UI.load(inf);
    }
}

void Window::loadGame(YAML::Node node)
{
    if(!node) return ;

    std::string path = node["file"].as<std::string>();
    float x = 0, y = 0, w = 1, h = 1;

    if(node["x"]) x = node["x"].as<float>() / 100;
    if(node["y"]) y = node["y"].as<float>() / 100;
    if(node["w"]) w = node["w"].as<float>() / 100;
    if(node["h"]) h = node["h"].as<float>() / 100;


    Interface* inf = new Game(UI.getRootFrame(), {x, y, w, h});
    inf->linkContent(path);
    UI.load(inf);
}
```

8.62 src/button/src/destructor.cpp File Reference

```
#include <button.hpp>
```

8.63 destructor.cpp

Go to the documentation of this file.

```
#include <button.hpp>

ButtonImage:: ~ButtonImage()
{
    for(auto &action : actions)
    {
        delete action;
    }
}
```

8.64 src/chunk/src/destructor.cpp File Reference

```
#include <chunk.hpp>
```

8.65 destructor.cpp

Go to the documentation of this file.

```
#include <chunk.hpp>

Chunk::~Chunk()
{
    for(auto i : visiter)
        delete i;

    while(!Entity.empty())
    {
        delete Entity.back();
        Entity.pop_back();
    }
}

```

8.66 src/container/src/destructor.cpp File Reference

```
#include <container.hpp>
```

8.67 destructor.cpp

Go to the documentation of this file.

```
#include <container.hpp>

Container::~Container()
{
    for(Sprite & sprite : sprites)
    {
        for(auto& frame : sprite)
            delete frame;
        sprite.clear();
    }
}
```

8.68 src/frame/src/destructor.cpp File Reference

```
#include <frame.hpp>
```

8.69 destructor.cpp

Go to the documentation of this file.

```
#include <frame.hpp>

Frame::~Frame()
{
}
```

8.70 src/game/src/destructor.cpp File Reference

```
#include <game.hpp>
```

8.71 destructor.cpp

Go to the documentation of this file.

```
#include <game.hpp>


Game::~Game()
{
    for(auto &i : cache)
    {
        delete i;
    }

    for(auto &i : chunks)
    {
        delete i;
    }
}
```

8.72 src/interface/src/destructor.cpp File Reference

```
#include <interface.hpp>
```

8.73 destructor.cpp

Go to the documentation of this file.

```
#include <interface.hpp>


Interface::~Interface()
{
    for (auto& i : containers)
        delete i;
    containers.clear();

    for (auto& i : nested)
        delete i;
    nested.clear();

    for (auto& i : keystrokes)
        delete i;
}
```

8.74 src/object/src/destructor.cpp File Reference

```
#include <object.hpp>
```

8.75 destructor.cpp

Go to the documentation of this file.

```
#include <object.hpp>


Object::~Object()
{
    for (auto &stroke : strokes)
    {
        delete stroke.stroke;
    }
    strokes.clear();

}
```

8.76 src/visual/src/destructor.cpp File Reference

```
#include <visual.hpp>
```

Functions

- void deleteSprite (Sprite sprite)
- void deleteSprites (std::vector< Sprite > *&sprites)

8.76.1 Function Documentation

8.76.1.1 deleteSprite()

```
void deleteSprite (
            Sprite sprite )
```

Definition at line 17 of file destructor.cpp.

```
00018 {
00019     for(auto& frame : sprite)
00020         delete frame;
00021     sprite.clear();
00022 }
```

8.76.1.2 deleteSprites()

```
void deleteSprites (
            std::vector< Sprite > *& sprites )
```

Definition at line 24 of file destructor.cpp.

```
00025 {
00026     for(Sprite & sprite : *sprites)
00027     {
00028         deleteSprite(sprite);
00029     }
00030     delete sprites;
00031 }
```

8.77 destructor.cpp

Go to the documentation of this file.

```
#include <visual.hpp>

void Visual::deleteTexture2D(Texture2D*& texture)
{
    if(texture != nullptr)
    {
        UnloadTexture(*texture);
        texture = nullptr;
    }
}

Visual::~Visual()
{
    m_texture.reset();
}

void deleteSprite(Sprite sprite)
{
    for(auto& frame : sprite)
        delete frame;
    sprite.clear();
}

void deleteSprites(std::vector<Sprite>*& sprites)
{
    for(Sprite & sprite : *sprites)
    {
        deleteSprite(sprite);
    }
    delete sprites;
}
```

8.78 src/window/src/destructor.cpp File Reference

```
#include <window.hpp>
```

8.79 destructor.cpp

Go to the documentation of this file.

```
#include <window.hpp>

Window::~Window()
{
    CloseWindow();
}
```

8.80 src/chunk/include/chunk.hpp File Reference

```
#include "action.hpp"
#include <chrono>
#include <deque>
#include <interface.hpp>
```

Classes

- class Chunk

  manages the spawning of chunks and how entities interact witht them
- class moveEntityAction

8.81 chunk.hpp

Go to the documentation of this file.

```
#ifndef CHUNK_HPP
#define CHUNK_HPP

#include "action.hpp"
#include <chrono>
#include <deque>

#include <interface.hpp>

class Chunk : public Interface
{
private:
    friend class moveEntityAction;
    fPoint velocity;
    std::vector<Container*> visiter;
    std::deque<Container*> Entity;
    std::chrono::time_point<std::chrono::system_clock> spawnClock;
    std::chrono::time_point<std::chrono::system_clock> moveClock;
    constexpr static std::chrono::duration<double> spawnTime = std::chrono::duration<double>(1.0);
    constexpr static std::chrono::duration<double> moveTime = std::chrono::duration<double>(0.1);

protected:
    void drawEntity();
    Container* randomEntity();
    void movingEntity();
public:
    Chunk(Frame*, Rectangle);
    Chunk(Chunk*);
    Chunk(Chunk*, Rectangle);
    Chunk(Chunk*, Frame*, Rectangle);
    ~Chunk();

    void addVisiter(Container*);
    void addVisiter(Container*, int);
    void addVisiter(Container*, Rectangle);
    void addVisiter(Container*, int, Rectangle);
    void generateEntity();

    void setVelocity(fPoint);

    std::string linkContent(std::string path) override;
    Action* getRuntimeEvent() override;

    void draw() override;
};

class moveEntityAction : public Action
{
private:
    Chunk* chunk;
public:
    moveEntityAction(Chunk*);
    ~moveEntityAction();

    void execute() override;
    Action* clone() override;
};
#endif

```

8.82 src/chunk/src/action/moveentity.cpp File Reference

```
#include <chunk.hpp>
```

8.83 moveentity.cpp

Go to the documentation of this file.

```
#include <chunk.hpp>
```

```

moveEntityAction::moveEntityAction(Chunk* chunk) : chunk(chunk)
{
}

moveEntityAction::~moveEntityAction()
{
}

void moveEntityAction::execute()
{
    chunk->movingEntity();
}

Action* moveEntityAction::clone()
{
    return new moveEntityAction(chunk);
}
```

8.84 src/container/include/container.hpp File Reference

```
#include <vector>
#include <visual.hpp>
#include <frame.hpp>
#include <action.hpp>
#include <const/datatype.hpp>
#include <const/path/atb.hpp>
#include <file.hpp>
```

Classes

- class Container

  holds specific entities and their behavior
- class changeImageAction

  changes display image of container

8.85 container.hpp

Go to the documentation of this file.

```
#ifndef CONTAINER_HPP
#define CONTAINER_HPP

#include <vector>

#include <visual.hpp>
#include <frame.hpp>
#include <action.hpp>
#include <const/datatype.hpp>
#include <const/path/atb.hpp>
#include <file.hpp>

class Container : public Frame
{
private:
    friend class changeImageAction;
    static int id_count;
    int instance_id;
    int probability;

    std::vector<Sprite> sprites;
    std::string name;
    iPoint focus;
    bool visible;

protected:
    bool loadName(YAML::Node node);
    void loadSprites(YAML::Node node);
    void loadFocus(YAML::Node node);
public:
    Container(Frame*, Rectangle);
    Container(Container*);
    Container(Container*, Rectangle);
    Container(Container*, Frame*, Rectangle);
    virtual ~Container();

    virtual std::string linkContent(std::string);
    virtual std::string linkContentAbsolute(std::string);
    std::string getName();

    void setProbability(int);
    int getProbability();

    void chooseSprite(int);

    void chooseImage(int);

    void chooseImage(int, int);

    void nextImage();

    void prevImage();

    void nextSprite();

    void prevSprite();

    bool isOverlapping(fPoint);
    bool isOverlapping(Rectangle);
    bool isOverlapping(Container*);
    float OverlappingArea(Rectangle);
    float OverlappingArea(Container*);

    virtual void draw();
    void show();
    void hide();
    void toggleVisibility();
    bool isVisible();
    int getInstanceId();

    virtual Action* react();
    virtual Action* getRuntimeEvent();
};

class changeImageAction : public Action
{
private:
    Container* container;
    iPoint focus;
public:
    changeImageAction(Container*, iPoint);
    changeImageAction(changeImageAction*);
    ~changeImageAction();
    void execute() override;
    Action* clone() override;
};
#endif
```

8.86 src/container/src/action/changesprite.cpp File Reference

`#include <container.hpp>`

8.87 changesprite.cpp

Go to the documentation of this file.

```
#include <container.hpp>

changeImageAction::changeImageAction(Container* c, iPoint p)
{
    container = c;
    focus = p;
}

changeImageAction::changeImageAction(changeImageAction* c)
{
    container = c->container;
    focus = c->focus;
}

changeImageAction::~changeImageAction()
{
    container = nullptr;
}

void changeImageAction::execute()
{
    container->chooseImage(focus[0], focus[1]);
}

Action* changeImageAction::clone()
{
    return new changeImageAction(this);
}
```

8.88 src/container/src/overlap.cpp File Reference

```
#include <container.hpp>
```

8.89 overlap.cpp

Go to the documentation of this file.

```
#include <container.hpp>

bool Container::isOverlapping(fPoint point)
{
    Rectangle rec = getFrame();
    return (point[0] >= rec.x && point[0] <= rec.x + rec.width && point[1] >= rec.y && point[1] <=
rec.y + rec.height);

}

bool Container::isOverlapping(Rectangle rec)
{
    Rectangle rec2 = getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
rec2.height && rec.y + rec.height >= rec2.y);
}

bool Container::isOverlapping(Container* container)
{
    Rectangle rec = getFrame();
    Rectangle rec2 = container->getFrame();
    return (rec.x <= rec2.x + rec2.width && rec.x + rec.width >= rec2.x && rec.y <= rec2.y +
rec2.height && rec.y + rec.height >= rec2.y);
}

float Container::OverlappingArea(Rectangle rec)
{
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}

float Container::OverlappingArea(Container* container)
{
    Rectangle rec = container->getFrame();
    Rectangle rec2 = getFrame();
    float x = std::max(rec.x, rec2.x);
    float y = std::max(rec.y, rec2.y);
    float w = std::min(rec.x + rec.width, rec2.x + rec2.width) - x;
    float h = std::min(rec.y + rec.height, rec2.y + rec2.height) - y;
    if(w < 0 || h < 0) return 0;
    return w * h;
}
```

8.90 src/frame/include/frame.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <mutex>
#include <raylib.h>
#include <const/datatype.hpp>
```

Classes

- class Frame

  position and size of object on screen

8.91 frame.hpp

Go to the documentation of this file.

```
#ifndef FRAME_HPP
#define FRAME_HPP

#include <iostream>
#include <vector>
#include <string>
#include <mutex>

#include <raylib.h>

#include <const/datatype.hpp>

class Frame
{
private:
    Rectangle frame;
    std::vector<Frame*> subframes;
    Frame* parent;

    fRect relative;

    mutable std::mutex mtx;
protected:
    virtual void updateFrame(bool recursive = false);
    bool isroot() const;
    void addSubframe(Frame* subframe);
    void removeSubframe(Frame* subframe);

    void beginUpdate();
    void endUpdate();
public:
    Frame(Frame* par, Rectangle rel);
    Frame(Frame* self);
    Frame(Rectangle rec);
    ~Frame();

    void plug(Frame* par, fRect rel);
    void plug(Frame* par);
    void unplug();

    void moveTo(fPoint rel);
    void moveTo(int x, int y);

    void moveCenterTo(fPoint rel);
    void moveCenterTo(int x, int y);

    void moveBy(fPoint rel);
    void moveBy(int, int);

    void resize(fPoint rel);
    void resize(int w, int h);

    const Rectangle& getFrame() const;
    const fRect& getRelative() const;
    Frame* getParent();

    void setRelative(fRect rel);

    const fPoint& getCenter() const;

    const float& getX() const;
    const float& getY() const;
    const float& getW() const;
    const float& getH() const;

    operator Rectangle() const;
    operator fRect() const;
    operator iRect() const;

};

#endif
```

8.92 src/frame/src/family.cpp File Reference

```
#include <frame.hpp>
#include <algorithm>
```

8.93 family.cpp

Go to the documentation of this file.

```
#include <frame.hpp>
#include <algorithm>


void Frame::plug(Frame* par, fRect rel)
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par, fRect rel): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    relative = rel;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}

void Frame::plug(Frame* par)
{
    if(par == nullptr)
    {
        throw std::runtime_error("Frame::plug(Frame* par): par is nullptr");
        return ;
    }
    mtx.lock();
    parent = par;
    mtx.unlock();
    updateFrame();

    parent->addSubframe(this);
}

void Frame::unplug()
```

```
{
    if(isroot()) return ;
    mtx.lock();
    parent->removeSubframe(this);
    parent = nullptr;
    mtx.unlock();
}

void Frame::addSubframe(Frame* subframe)
{
    mtx.lock();
    subframes.push_back(subframe);
    mtx.unlock();
}

void Frame::removeSubframe(Frame* subframe)
{
    mtx.lock();
    int i = subframes.size() - 1;
    while(i >= 0 && subframes.size())
    {
        while(!subframes.empty() && subframes.back() == subframe)
            subframes.pop_back();
        i = std::min(i, (int) subframes.size() - 1);
        if(!subframes.empty() && subframes[i] == subframe)
        {
            subframes[i] = subframes.back();
            subframes.pop_back();
        }
    }
    mtx.unlock();
}

bool Frame::isroot() const
{
    std::lock_guard<std::mutex> lock(mtx);
    return parent == nullptr;
}

void Frame::beginUpdate()
{
    mtx.lock();
}

void Frame::endUpdate()
{
    mtx.unlock();
}
```

8.94 src/game/include/game.hpp File Reference

```
#include "action.hpp"
#include <deque>
#include <chrono>
#include <raylib.h>
#include <frame.hpp>
#include <container.hpp>
#include <keystroke.hpp>
#include <interface.hpp>
#include <chunk.hpp>
```

Classes

- class Game
- class moveChunksAction

8.95 game.hpp

Go to the documentation of this file.

```
#ifndef GAME_HPP
#define GAME_HPP

#include "action.hpp"
#include <deque>
#include <chrono>

#include <raylib.h>

#include <frame.hpp>
#include <container.hpp>
#include <keystroke.hpp>
#include <interface.hpp>
#include <chunk.hpp>

class Game : public Interface
{
private:
    friend class moveChunksAction;
    std::deque<Chunk*> chunks;
    std::vector<Chunk*> cache;
    Container* main;
    fPoint mapDisplacement;
    fPoint mapDirection;
    float mapSpeed;
    std::chrono::time_point<std::chrono::system_clock> mapSpeedClock;
    bool initState;
protected:
    void loadChunk(YAML::Node);
    void loadCollide(YAML::Node);
    void loadEvent(YAML::Node);
    void loadAttactObject(YAML::Node);
    void loadMap();
public:
    Game(Frame*, Rectangle);
    Game(Game*);
    Game(Game*, Rectangle);
    Game(Game*, Frame*, Rectangle);

    ~Game();

    std::string linkContentAbsolute(std::string path) override;

    Action* react() override;
    Action* getRuntimeEvent() override;
    void draw() override;

};

class moveChunksAction : public Action
{
private:
    Game* game;
    fPoint delta;
    fPoint direction;
    float speed;
public:
    moveChunksAction(Game*, fPoint);
    moveChunksAction(Game*, fPoint, float);
    ~moveChunksAction();

    void execute() override;
    Action* clone() override;
};
#endif
```

8.96 src/game/src/action/movechunk.cpp File Reference

```
#include <game.hpp>
#include <vector.hpp>
```

8.97 movechunk.cpp

Go to the documentation of this file.
```
#include <game.hpp>
#include <vector.hpp>

moveChunksAction::moveChunksAction(Game* game, fPoint delta)
{
    this->game = game;
    this->delta = delta;
}

moveChunksAction::moveChunksAction(Game* game, fPoint d, float v)
{
    this->game = game;
    this->direction = d;
    this->speed = v;

    float angle = VECTOR2D::getAngle(direction);
    delta[0] = cos(angle) * speed;
    delta[1] = sin(angle) * speed;
}

moveChunksAction::~moveChunksAction()
{
}

void moveChunksAction::execute()
{
    for(auto& chunk : game->chunks)
    {
        chunk->moveBy(delta);
    }
    game->loadMap();
}

Action* moveChunksAction::clone()
{
    return new moveChunksAction(game, delta);
}
```

8.98 src/interface/include/interface.hpp File Reference

```
#include "action.hpp"
#include <raylib.h>
#include <frame.hpp>
#include <container.hpp>
#include <keystroke.hpp>
#include <button.hpp>
```

Classes

- class Interface

  where user can interact with the game
- class moveObjectAction

  manages the features of a movement, including which object, speed, direction etc.

8.99 interface.hpp

Go to the documentation of this file.
```
#ifndef INTERFACE_HPP
#define INTERFACE_HPP

```

```
#include "action.hpp"
#include <raylib.h>

#include <frame.hpp>
#include <container.hpp>
#include <keystroke.hpp>
#include <button.hpp>

class Interface : public Container
{
private:
    friend class moveObjectAction;

    std::vector<Container*> containers;
    std::vector<Interface*> nested;
    std::vector<KeyStroke*> keystrokes;
protected:
    void loadObject(YAML::Node);
    void loadControl(YAML::Node);
    void loadButton(YAML::Node);
    void drawNested();
    void drawContainers();

public:
    Interface(Frame*, Rectangle);
    Interface(Interface*);
    Interface(Interface*, Rectangle);
    Interface(Interface*, Frame*, Rectangle);

    ~Interface();

    Container* getContainers(int);
    int getContainersSize();

    std::string linkContent(std::string path) override;
    std::string linkContentAbsolute(std::string path) override;

    Action* react() override;
    Action* getRuntimeEvent() override;
    void draw() override;
};

class moveObjectAction : public Action
{
private:
    Container* obj;
    fPoint delta;
    fPoint dir;
    float speed;
public:
    moveObjectAction(Container* obj, fPoint delta);
    moveObjectAction(Container* obj, fPoint dir, float speed);
    ~moveObjectAction();

    void execute() override;
    Action* clone() override;
};
#endif
```

8.100 src/interface/src/action/moveobject.cpp File Reference

```
#include <interface.hpp>
#include <vector.hpp>
```

8.101 moveobject.cpp

Go to the documentation of this file.

```
#include <interface.hpp>
#include <vector.hpp>

moveObjectAction::moveObjectAction(Container* obj, fPoint delta) : obj(obj), delta(delta)
{
}
```

```
00007
00008 moveObjectAction::moveObjectAction(Container* obj, fPoint dir, float speed) : obj(obj), dir(dir),
      speed(speed)
00009 {
00010     float angle = VECTOR2D::getAngle(dir);
00011     delta[0] = cos(angle) * speed;
00012     delta[1] = sin(angle) * speed;
00013 }
00014
00015 moveObjectAction::~moveObjectAction()
00016 {
00017 }
00018
00019 void moveObjectAction::execute()
00020 {
00021     obj->moveBy(delta);
00022 }
00023
00024
00025 Action* moveObjectAction::clone()
00026 {
00027     return new moveObjectAction(obj, delta);
00028 }
```

8.102 src/main.cpp File Reference

```
#include <iostream>
#include <window.hpp>
```

Functions

- int main ()

8.102.1 Function Documentation

8.102.1.1 main()

```
int main ( )
```

Definition at line 5 of file main.cpp.

```
00006 {
00007     Window win("window.yaml");
00008     win.run();
00009     return 0;
00010 }
```

8.103 main.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002
00003 #include <window.hpp>
00004
00005 int main()
00006 {
00007     Window win("window.yaml");
00008     win.run();
00009     return 0;
00010 }
```

8.104 src/object/include/object.hpp File Reference

```
#include "action.hpp"
#include <container.hpp>
#include <keystroke.hpp>
#include <chrono>
```

Classes

- class Object

8.105 object.hpp

Go to the documentation of this file.

```
#ifndef OBJECT_HPP
#define OBJECT_HPP

#include "action.hpp"
#include <container.hpp>
#include <keystroke.hpp>
#include <chrono>
class Object : public Container
{
private:
    struct ObjectKeyStroke
    {
        KeyStroke* stroke;
    };
    std::vector<ObjectKeyStroke> strokes;
    std::chrono::time_point<std::chrono::steady_clock> waitUntil;


protected:
    void loadControl(YAML::Node node);

public:
    Object(Frame*, Rectangle);
    Object(Object*);
    Object(Object*, Rectangle);
    Object(Object*, Frame*, Rectangle);
    ~Object();

    virtual std::string linkContent(std::string) override;
    virtual std::string linkContentAbsolute(std::string) override;

    virtual Action* react() override;
    void draw() override;
};

#endif
```

8.106 src/utils/include/const/datatype.hpp File Reference

```
#include <array>
#include <vector>
#include <string>
```

Typedefs

- using iPoint = std::array< int, 2 >
- using fPoint = std::array< float, 2 >
- using iRect = std::array< int, 4 >
- using fRect = std::array< float, 4 >
- using vi = std::vector< int >
- using vf = std::vector< float >

8.106.1 Typedef Documentation

8.106.1.1 fPoint

```
using fPoint = std::array<float, 2>
```

Definition at line 9 of file datatype.hpp.

8.106.1.2 fRect

```
using fRect = std::array<float, 4>
```

Definition at line 12 of file datatype.hpp.

8.106.1.3 iPoint

```
using iPoint = std::array<int, 2>
```

Definition at line 8 of file datatype.hpp.

8.106.1.4 iRect

```
using iRect = std::array<int, 4>
```

Definition at line 11 of file datatype.hpp.

8.106.1.5 vf

```
using vf = std::vector<float>
```

Definition at line 15 of file datatype.hpp.

8.106.1.6 vi

```
using vi = std::vector<int>
```

Definition at line 14 of file datatype.hpp.

8.107 datatype.hpp

Go to the documentation of this file.

```
#ifndef CONSTANT_HPP
#define CONSTANT_HPP

#include <array>
#include <vector>
#include <string>

using iPoint = std::array<int, 2>;
using fPoint = std::array<float, 2>;

using iRect = std::array<int, 4>;
using fRect = std::array<float, 4>;

using vi = std::vector<int>;
using vf = std::vector<float>;


#endif
```

8.108 src/utils/include/const/path/assets.hpp File Reference

```
#include <string>
```

Namespaces

- namespace PASSETS

Variables

- const std::string PASSETS::GRAPHIC_ = "assets/graphics/"
- const std::string PASSETS::SOUND_ = "assets/sounds/"
- const std::string PASSETS::FONT_ = "assets/fonts/"

8.109 assets.hpp

Go to the documentation of this file.

```
#ifndef ASSETS_HPP
#define ASSETS_HPP

#include <string>

namespace PASSETS
{
    extern const std::string GRAPHIC_;
    extern const std::string SOUND_;
    extern const std::string FONT_;
}

#endif
```

8.110 src/utils/include/const/path/atb.hpp File Reference

```
#include <string>
```

Namespaces

- namespace PATB

Variables

- const std::string PATB::ATB_ = "atb/"
- const std::string PATB::WINDOW_ = "atb/window/"
- const std::string PATB::INTERFACE_ = "atb/interface/"
- const std::string PATB::BUTTON_ = "atb/button/"
- const std::string PATB::CONTAINER_ = "atb/container/"
- const std::string PATB::OBJECT_ = "atb/object/"
- const std::string PATB::MAP_ = "atb/map/"
- const std::string PATB::CHUNK_ = "atb/chunk/"
- const std::string PATB::BLOCK_ = "atb/block/"
- const std::string PATB::ENTITY_ = "atb/entity/"

8.111 atb.hpp

Go to the documentation of this file.

```
#ifndef ATB_HPP
#define ATB_HPP

#include <string>

namespace PATB
{
    extern const std::string ATB_;

    extern const std::string WINDOW_;

    extern const std::string INTERFACE_;
    extern const std::string BUTTON_;
    extern const std::string CONTAINER_;
    extern const std::string OBJECT_;


    extern const std::string MAP_;
    extern const std::string CHUNK_;
    extern const std::string BLOCK_;
    extern const std::string ENTITY_;

}

#endif
```

8.112 src/utils/include/countdown.hpp File Reference

```
#include <chrono>
```

Classes

- class CountDown

  count the time a playthrough takes

8.113 countdown.hpp

Go to the documentation of this file.

```
#ifndef COUNT_DOWN_HPP
#define COUNT_DOWN_HPP

#include <chrono>

class CountDown
{
private:
    std::chrono::time_point<std::chrono::system_clock> start;
    std::chrono::duration<double> elapsed_seconds;
    bool finished;
public:
    CountDown(int milliseconds);
    ~CountDown();
    int get();
    bool isFinished();
    void run();
};

#endif
```

8.114 src/utils/include/file.hpp File Reference

```
#include <string>
#include <vector>
#include <yaml-cpp/yaml.h>
```

Namespaces

- namespace YAML_FILE

  opens and interacts with YAML files

Functions

- bool YAML_FILE::isFile (std::string path)
- YAML::Node YAML_FILE::readFile (std::string path)
- bool YAML_FILE::writeFile (std::string path, YAML::Node content)

8.115 file.hpp

Go to the documentation of this file.

```
#ifndef UTILS_FILE_H
#define UTILS_FILE_H

#include <string>
#include <vector>

#include <yaml-cpp/yaml.h>

namespace YAML_FILE
{
    bool isFile(std::string path);
    YAML::Node readFile(std::string path);
    bool writeFile(std::string path, YAML::Node content);
}

#endif
```

8.116 src/utils/include/keystroke.hpp File Reference

```
#include <action.hpp>
#include <vector>
#include <string>
```

Classes

- class KeyStroke

  manages the link between a key and the actions it performs

Functions

- int toKey (std::string)

8.116.1 Function Documentation

8.116.1.1 toKey()

```
int toKey (
            std::string x )
```

Definition at line 68 of file keystroke.cpp.

```
{
    if(x.size() == 1)
    {
        if(x[0] >= 'a' && x[0] <= 'z')
            return x[0] - 'a' + KEY_A;
        if(x[0] >= 'A' && x[0] <= 'Z')
            return x[0] - 'A' + KEY_A;

        if(x[0] >= '0' && x[0] <= '9')
            return x[0] - '0' + KEY_ZERO;

        switch (x[0]) {
            case ' ':
                return KEY_SPACE;
            case '.':
                return KEY_PERIOD;
            case ',':
                return KEY_COMMA;
            case ';':
                return KEY_SEMICOLON;
            case '\"':
                return KEY_APOSTROPHE;
            case '/':
                return KEY_SLASH;
            case '\\':
                return KEY_BACKSLASH;
            case '-':
                return KEY_MINUS;
            case '=':
                return KEY_EQUAL;
            case '[':
                return KEY_LEFT_BRACKET;
            case ']':
                return KEY_RIGHT_BRACKET;
            case '`':
                return KEY_GRAVE;
            case '~':
                return KEY_GRAVE;
            case '!':
                return KEY_ONE;
            case '@':
                return KEY_ONE;
            case '#':
                return KEY_THREE;
            case '$':
                return KEY_FOUR;
            case '%':
                return KEY_FIVE;
            case '^':
                return KEY_SIX;
            case '&':
                return KEY_SEVEN;
            case '*':
                return KEY_EIGHT;
            case '(':
                return KEY_NINE;
            case ')':
                return KEY_ZERO;
            case '_':
                return KEY_MINUS;
            case '+':
                return KEY_EQUAL;
            case '{':
                return KEY_LEFT_BRACKET;
            case '}':
                return KEY_RIGHT_BRACKET;
            case ':':
                return KEY_SEMICOLON;
            case '"':
                return KEY_APOSTROPHE;
            case '<':
                return KEY_COMMA;
            case '>':
                return KEY_PERIOD;
            case '?':
                return KEY_SLASH;
        }
    }else
    {
        if(x == "esc") return KEY_ESCAPE; if(x == "enter") return KEY_ENTER;
        if(x == "tab") return KEY_TAB;

        if(x == "shift") return KEY_LEFT_SHIFT;
        if(x == "control") return KEY_LEFT_CONTROL;
        if(x == "alt") return KEY_LEFT_ALT;
        if(x == "super") return KEY_LEFT_SUPER;

        if(x == "right") return KEY_RIGHT;
        if(x == "left") return KEY_LEFT;
        if(x == "down") return KEY_DOWN;
        if(x == "up") return KEY_UP;

        if(x == "leftshift") return KEY_LEFT_SHIFT;
        if(x == "leftcontrol") return KEY_LEFT_CONTROL;
        if(x == "leftalt") return KEY_LEFT_ALT;
        if(x == "leftsuper") return KEY_LEFT_SUPER;
        if(x == "rightshift") return KEY_RIGHT_SHIFT;
        if(x == "rightcontrol") return KEY_RIGHT_CONTROL;
        if(x == "rightalt") return KEY_RIGHT_ALT;
        if(x == "rightsuper") return KEY_RIGHT_SUPER;
        if(x == "menu") return KEY_MENU;

        if(x == "backspace") return KEY_BACKSPACE;
        if(x == "insert") return KEY_INSERT;
        if(x == "delete") return KEY_DELETE;
        if(x == "pause") return KEY_PAUSE;

        if(x == "f1") return KEY_F1;
        if(x == "f2") return KEY_F2;
        if(x == "f3") return KEY_F3;
        if(x == "f4") return KEY_F4;
        if(x == "f5") return KEY_F5;
        if(x == "f6") return KEY_F6;
        if(x == "f7") return KEY_F7;
        if(x == "f8") return KEY_F8;
        if(x == "f9") return KEY_F9;
        if(x == "f10") return KEY_F10;
        if(x == "f11") return KEY_F11;
        if(x == "f12") return KEY_F12;

        if(x == "pageup") return KEY_PAGE_UP;
        if(x == "pagedown") return KEY_PAGE_DOWN;
        if(x == "home") return KEY_HOME;
        if(x == "end") return KEY_END;
        if(x == "capslock") return KEY_CAPS_LOCK;
        if(x == "scrolllock") return KEY_SCROLL_LOCK;
        if(x == "numlock") return KEY_NUM_LOCK;
        if(x == "printscreen") return KEY_PRINT_SCREEN;
    }

    return 0;
}
```

8.117 keystroke.hpp

Go to the documentation of this file.

```
#ifndef KEYSTROKE_HPP
#define KEYSTROKE_HPP

#include <action.hpp>
#include <vector>
#include <string>

int toKey(std::string);

class KeyStroke
{
private:
    std::vector<int> key;
    std::vector< Action* > action;
    int id;
public:
    KeyStroke();
    KeyStroke(std::vector<int>);
    ~KeyStroke();

    int size();
    void add(unsigned char);
    void setAction(std::vector<Action*>);
    void addAction(Action*);
    void chooseAction(int);
    int getCurrent(int);
    void nextAction();

    Action* react();
};

#endif
```

8.118 src/utils/include/random.hpp File Reference

```
#include <random>
#include <string>
```

Classes

- class RandomEngine

8.119 random.hpp

Go to the documentation of this file.

```
#ifndef RANDOM_HPP
#define RANDOM_HPP

#include <random>
#include <string>

class RandomEngine
{
private:
    std::mt19937 engine;
public:
    RandomEngine();
    RandomEngine(unsigned int seed);
    ~RandomEngine();
    int randInt(int min = 0, int max = 1);
    double randDouble(double min = 0, double max = 1);
    char randChar(char min = 0, char max = 127);
    std::string randString(int length, char min, char max);
    std::string randInt2String(int length, int min = 0, int max = 9);
    std::string randString(int length, bool haveDigit = true, bool haveLower = true, bool haveUpper =
true, bool haveSpecial = true);
};


#endif
```

8.120 src/utils/include/vector.hpp File Reference

```
#include <const/datatype.hpp>
#include <math.h>
```

Namespaces

- namespace VECTOR2D

Functions

- float VECTOR2D::getAngle (fPoint v1)
- float VECTOR2D::getAngle (fPoint v1, fPoint v2)

8.121 vector.hpp

Go to the documentation of this file.

```
#ifndef MY_VECTOR_SPACE_HPP
#define MY_VECTOR_SPACE_HPP

#include <const/datatype.hpp>
#include <math.h>

namespace VECTOR2D
{
    float getAngle(fPoint v1);
    float getAngle(fPoint v1, fPoint v2);
};

#endif
```

8.122 src/utils/src/constant.cpp File Reference

```
#include <const/path/assets.hpp>
#include <const/path/atb.hpp>
```

8.123 constant.cpp

Go to the documentation of this file.

```
#include <const/path/assets.hpp>
#include <const/path/atb.hpp>

const std::string PASSETS::GRAPHIC_ = "assets/graphics/";
const std::string PASSETS::SOUND_ = "assets/sounds/";
const std::string PASSETS::FONT_ = "assets/fonts/";

const std::string PATB::ATB_ = "atb/";
const std::string PATB::WINDOW_ = "atb/window/";
const std::string PATB::INTERFACE_ = "atb/interface/";
const std::string PATB::BUTTON_ = "atb/button/";
const std::string PATB::CONTAINER_ = "atb/container/";
const std::string PATB::OBJECT_ = "atb/object/";
const std::string PATB::MAP_ = "atb/map/";
const std::string PATB::CHUNK_ = "atb/chunk/";
const std::string PATB::BLOCK_ = "atb/block/";
const std::string PATB::ENTITY_ = "atb/entity/";

```

8.124 src/utils/src/countdown.cpp File Reference

```
#include <countdown.hpp>
```

8.125 countdown.cpp

Go to the documentation of this file.

```
#include <countdown.hpp>

CountDown::CountDown(int milliseconds)
{
    start = std::chrono::system_clock::now();
    finished = false;
    elapsed_seconds = std::chrono::milliseconds(milliseconds);
}

CountDown::~CountDown()
{
}

bool CountDown::isFinished()
{
    return finished || (std::chrono::system_clock::now() - start) > elapsed_seconds;
}

void CountDown::run()
{
    finished = false;
    start = std::chrono::system_clock::now();
}

int CountDown::get()
{
    return elapsed_seconds.count() * 1000;
}
```

8.126 src/utils/src/file.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <file.hpp>
```

8.127 file.cpp

Go to the documentation of this file.

```
#include <iostream>
#include <fstream>
#include <file.hpp>

bool YAML_FILE::isFile(std::string path)
{
    // return true if file exists

    std::ifstream fin(path);
    return fin.good();
}

YAML::Node YAML_FILE::readFile(std::string path)
{
    // return YAML::Node from file

    YAML::Node node;
    try
    {
        node = YAML::LoadFile(path);
    }
    catch (YAML::BadFile& e)
    {
        std::cout « "Error: " « e.what() « std::endl;
    }
    return node;
}
```

8.128 src/utils/src/keystroke.cpp File Reference

```
#include <keystroke.hpp>
#include <raylib.h>
#include <string>
#include <iostream>
```

Functions

- int toKey (std::string x)

8.128.1 Function Documentation

8.128.1.1 toKey()

```
int toKey (
            std::string x )
```

Definition at line 68 of file keystroke.cpp.

```
{
    if(x.size() == 1)
    {
        if(x[0] >= 'a' && x[0] <= 'z')
            return x[0] - 'a' + KEY_A;
        if(x[0] >= 'A' && x[0] <= 'Z')
            return x[0] - 'A' + KEY_A;

        if(x[0] >= '0' && x[0] <= '9')
            return x[0] - '0' + KEY_ZERO;

        switch (x[0]) {
            case ' ':
                return KEY_SPACE;
            case '.':
                return KEY_PERIOD;
            case ',':
                return KEY_COMMA;
            case ';':
                return KEY_SEMICOLON;
            case '\'':
                return KEY_APOSTROPHE;
            case '/':
                return KEY_SLASH;
            case '\\':
                return KEY_BACKSLASH;
            case '-':
                return KEY_MINUS;
            case '=':
                return KEY_EQUAL;
            case '[':
                return KEY_LEFT_BRACKET;
            case ']':
                return KEY_RIGHT_BRACKET;
            case '`':
                return KEY_GRAVE;
            case '~':
                return KEY_GRAVE;
            case '!':
                return KEY_ONE;
            case '@':
                return KEY_ONE;
            case '#':
                return KEY_THREE;
            case '$':
                return KEY_FOUR;
            case '%':
                return KEY_FIVE;
            case '^':
                return KEY_SIX;
            case '&':
                return KEY_SEVEN;
            case '*':
                return KEY_EIGHT;
            case '(':
                return KEY_NINE;
            case ')':
                return KEY_ZERO;
            case '_':
                return KEY_MINUS;
            case '+':
                return KEY_EQUAL;
            case '{':
                return KEY_LEFT_BRACKET;
            case '}':
                return KEY_RIGHT_BRACKET;
            case ':':
                return KEY_SEMICOLON;
            case '"':
                return KEY_APOSTROPHE;
            case '<':
                return KEY_COMMA;
            case '>':
                return KEY_PERIOD;
            case '?':
                return KEY_SLASH;
        }
    }else
    {
        if(x == "esc") return KEY_ESCAPE; if(x == "enter") return KEY_ENTER;
        if(x == "tab") return KEY_TAB;

        if(x == "shift") return KEY_LEFT_SHIFT;
        if(x == "control") return KEY_LEFT_CONTROL;
        if(x == "alt") return KEY_LEFT_ALT;
        if(x == "super") return KEY_LEFT_SUPER;

        if(x == "right") return KEY_RIGHT;
        if(x == "left") return KEY_LEFT;
        if(x == "down") return KEY_DOWN;
        if(x == "up") return KEY_UP;

        if(x == "leftshift") return KEY_LEFT_SHIFT;
        if(x == "leftcontrol") return KEY_LEFT_CONTROL;
        if(x == "leftalt") return KEY_LEFT_ALT;
        if(x == "leftsuper") return KEY_LEFT_SUPER;
        if(x == "rightshift") return KEY_RIGHT_SHIFT;
        if(x == "rightcontrol") return KEY_RIGHT_CONTROL;
        if(x == "rightalt") return KEY_RIGHT_ALT;
        if(x == "rightsuper") return KEY_RIGHT_SUPER;
        if(x == "menu") return KEY_MENU;

        if(x == "backspace") return KEY_BACKSPACE;
        if(x == "insert") return KEY_INSERT;
        if(x == "delete") return KEY_DELETE;
        if(x == "pause") return KEY_PAUSE;

        if(x == "f1") return KEY_F1;
        if(x == "f2") return KEY_F2;
        if(x == "f3") return KEY_F3;
        if(x == "f4") return KEY_F4;
        if(x == "f5") return KEY_F5;
        if(x == "f6") return KEY_F6;
        if(x == "f7") return KEY_F7;
        if(x == "f8") return KEY_F8;
        if(x == "f9") return KEY_F9;
        if(x == "f10") return KEY_F10;
        if(x == "f11") return KEY_F11;
        if(x == "f12") return KEY_F12;

        if(x == "pageup") return KEY_PAGE_UP;
        if(x == "pagedown") return KEY_PAGE_DOWN;
        if(x == "home") return KEY_HOME;
        if(x == "end") return KEY_END;
        if(x == "capslock") return KEY_CAPS_LOCK;
        if(x == "scrolllock") return KEY_SCROLL_LOCK;
        if(x == "numlock") return KEY_NUM_LOCK;
        if(x == "printscreen") return KEY_PRINT_SCREEN;
    }

    return 0;
}
```

8.129 keystroke.cpp

Go to the documentation of this file.

```
#include <keystroke.hpp>
#include <raylib.h>
#include <string>
#include <iostream>

KeyStroke::KeyStroke()
{
    id = 0;
}

KeyStroke::KeyStroke(std::vector<int> k)
{
    key = k;
    id = 0;
}

KeyStroke::~KeyStroke()
{
    for(auto &a : action)
    {
        delete a;
    }
}

int KeyStroke::size()
{
    return key.size();
}

void KeyStroke::add(unsigned char k)
{
    key.push_back(k);
}
void KeyStroke::setAction(std::vector<Action*> a)
{
    action = a;
}

void KeyStroke::addAction(Action* a)
{
    action.push_back(a);
}

Action* KeyStroke::react()
{
    for(auto k : key)
    {
        if(!IsKeyDown(k)) return nullptr;
    }
    return action[id]->clone();
}
void KeyStroke::chooseAction(int i)
{
    id = i;
}

int KeyStroke::getCurrent(int i)
{
    return id;
}

void KeyStroke::nextAction()
{
    id = (id + 1) % action.size();
}


int toKey(std::string x)
{
    if(x.size() == 1)
    {
        if(x[0] >= 'a' && x[0] <= 'z')
            return x[0] - 'a' + KEY_A;
        if(x[0] >= 'A' && x[0] <= 'Z')
            return x[0] - 'A' + KEY_A;

        if(x[0] >= '0' && x[0] <= '9')
            return x[0] - '0' + KEY_ZERO;

        switch (x[0]) {
            case ' ':
                return KEY_SPACE;
            case '.':
                return KEY_PERIOD;
            case ',':
                return KEY_COMMA;
            case ';':
                return KEY_SEMICOLON;
            case '\'':
                return KEY_APOSTROPHE;
            case '/':
                return KEY_SLASH;
            case '\\':
                return KEY_BACKSLASH;
            case '-':
                return KEY_MINUS;
            case '=':
                return KEY_EQUAL;
            case '[':
                return KEY_LEFT_BRACKET;
            case ']':
                return KEY_RIGHT_BRACKET;
            case '`':
                return KEY_GRAVE;
            case '~':
                return KEY_GRAVE;
            case '!':
                return KEY_ONE;
            case '@':
                return KEY_ONE;
            case '#':
                return KEY_THREE;
            case '$':
                return KEY_FOUR;
            case '%':
                return KEY_FIVE;
            case '^':
                return KEY_SIX;
            case '&':
                return KEY_SEVEN;
            case '*':
                return KEY_EIGHT;
            case '(':
                return KEY_NINE;
            case ')':
                return KEY_ZERO;
            case '_':
                return KEY_MINUS;
            case '+':
                return KEY_EQUAL;
            case '{':
                return KEY_LEFT_BRACKET;
            case '}':
                return KEY_RIGHT_BRACKET;
            case ':':
                return KEY_SEMICOLON;
            case '"':
                return KEY_APOSTROPHE;
            case '<':
                return KEY_COMMA;
            case '>':
                return KEY_PERIOD;
            case '?':
                return KEY_SLASH;
        }
    }else
    {
        if(x == "esc") return KEY_ESCAPE; if(x == "enter") return KEY_ENTER;
        if(x == "tab") return KEY_TAB;

        if(x == "shift") return KEY_LEFT_SHIFT;
        if(x == "control") return KEY_LEFT_CONTROL;
        if(x == "alt") return KEY_LEFT_ALT;
        if(x == "super") return KEY_LEFT_SUPER;

        if(x == "right") return KEY_RIGHT;
        if(x == "left") return KEY_LEFT;
        if(x == "down") return KEY_DOWN;
        if(x == "up") return KEY_UP;

        if(x == "leftshift") return KEY_LEFT_SHIFT;
        if(x == "leftcontrol") return KEY_LEFT_CONTROL;
        if(x == "leftalt") return KEY_LEFT_ALT;
        if(x == "leftsuper") return KEY_LEFT_SUPER;
        if(x == "rightshift") return KEY_RIGHT_SHIFT;
        if(x == "rightcontrol") return KEY_RIGHT_CONTROL;
        if(x == "rightalt") return KEY_RIGHT_ALT;
        if(x == "rightsuper") return KEY_RIGHT_SUPER;
        if(x == "menu") return KEY_MENU;

        if(x == "backspace") return KEY_BACKSPACE;
        if(x == "insert") return KEY_INSERT;
        if(x == "delete") return KEY_DELETE;
        if(x == "pause") return KEY_PAUSE;

        if(x == "f1") return KEY_F1;
        if(x == "f2") return KEY_F2;
        if(x == "f3") return KEY_F3;
        if(x == "f4") return KEY_F4;
        if(x == "f5") return KEY_F5;
        if(x == "f6") return KEY_F6;
        if(x == "f7") return KEY_F7;
        if(x == "f8") return KEY_F8;
        if(x == "f9") return KEY_F9;
        if(x == "f10") return KEY_F10;
        if(x == "f11") return KEY_F11;
        if(x == "f12") return KEY_F12;

        if(x == "pageup") return KEY_PAGE_UP;
        if(x == "pagedown") return KEY_PAGE_DOWN;
        if(x == "home") return KEY_HOME;
        if(x == "end") return KEY_END;
        if(x == "capslock") return KEY_CAPS_LOCK;
        if(x == "scrolllock") return KEY_SCROLL_LOCK;
        if(x == "numlock") return KEY_NUM_LOCK;
        if(x == "printscreen") return KEY_PRINT_SCREEN;
    }

    return 0;
}
```

8.130 src/utils/src/random.cpp File Reference

```
#include <random.hpp>
#include <chrono>
```

8.131 random.cpp

Go to the documentation of this file.

```cpp
#include <random.hpp>

#include <chrono>


RandomEngine::RandomEngine()
{
    unsigned int seed = std::chrono::system_clock::now().time_since_epoch().count();
    engine.seed(seed);
}

RandomEngine::RandomEngine(unsigned int seed)
{
    engine.seed(seed);
}

RandomEngine::~RandomEngine()
{
}

int RandomEngine::randInt(int min, int max)
{
    std::uniform_int_distribution<int> distribution(min, max);
    return distribution(engine);
}

double RandomEngine::randDouble(double min, double max)
{
    std::uniform_real_distribution<double> distribution(min, max);
    return distribution(engine);
}

char RandomEngine::randChar(char min, char max)
{
    std::uniform_int_distribution<int> distribution(min, max);
    return distribution(engine);
}

std::string RandomEngine::randString(int length, char min, char max)
{
    std::string str;
    for (int i = 0; i < length; i++)
    {
        str += randChar(min, max);
    }
    return str;
}

std::string RandomEngine::randInt2String(int length, int min, int max)
{
    std::string str;
    for (int i = 0; i < length; i++)
    {
        str += std::to_string(randInt(min, max));
    }
    return str;
}

std::string RandomEngine::randString(int length, bool haveDigit, bool haveLower, bool haveUpper, bool
haveSpecial)
{
    std::string str;
    std::string digit = "0123456789";
    std::string lower = "abcdefghijklmnopqrstuvwxyz";
    std::string upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    std::string special = "!@#$%^&*()_+{}|:<>?~`-=[]\\;',./\"";
    std::string all = digit + lower + upper + special;
    if (haveDigit)
    {
        str += digit;
    }
    if (haveLower)
    {
        str += lower;
    }
    if (haveUpper)
    {
        str += upper;
    }
    if (haveSpecial)
    {
        str += special;
    }
    if (str.empty())
    {
        str = all;
    }
00087     std::string result;
00088     for (int i = 0; i < length; i++)
00089     {
00090         result += str[randInt(0, str.size() - 1)];
00091     }
00092     return result;
00093 }
```

8.132 src/utils/src/vector.cpp File Reference

```
#include <vector.hpp>
```

Functions

- float sqr (float x)

8.132.1 Function Documentation

8.132.1.1 sqr()

```
float sqr (
            float x )
```

Definition at line 11 of file vector.cpp.

```
00012 {
00013     return x * x;
00014 }
```

8.133 vector.cpp

Go to the documentation of this file.

```
00001 #include <vector.hpp>
00002 
00003 using namespace VECTOR2D;
00004 
00005 float VECTOR2D::getAngle(fPoint v1)
00006 {
00007     // arctan(y / x)
00008     return atan2(v1[1], v1[0]);
00009 }
00010 
00011 float sqr(float x)
00012 {
00013     return x * x;
00014 }
00015 
00016 float VECTOR2D::getAngle(fPoint v1, fPoint v2)
00017 {
00018     // angle between 2 vector
00019     // v1 * v2 = |v1| * |v2| * cos(angle)
00020 
00021     float dot = v1[0] * v2[0] + v1[1] * v2[1];
00022     float abs1 = sqrt(sqr(v1[0]) + sqr(v1[1]));
00023     float abs2 = sqrt(sqr(v2[0]) + sqr(v2[1]));
00024     return acos(dot / (abs1 * abs2));
00025 }
```

8.134 src/visual/include/visual.hpp File Reference

```
#include <memory>
#include <raylib.h>
#include <yaml-cpp/yaml.h>
#include <frame.hpp>
```

Classes

- class Visual

  images displayed on screen

Typedefs

- using Sprite = std::vector<Visual∗>

Functions

- void deleteSprite (Sprite ∗&)
- void deleteSprites (std::vector< Sprite > ∗&)

8.134.1 Typedef Documentation

8.134.1.1 Sprite

```
using Sprite = std::vector<Visual*>
```

Definition at line 36 of file visual.hpp.

8.134.2 Function Documentation

8.134.2.1 deleteSprite()

```
void deleteSprite (
            Sprite *&   )
```

8.134.2.2 deleteSprites()

```
void deleteSprites (
            std::vector< Sprite > *& sprites )
```

Definition at line 24 of file destructor.cpp.

```
{
    for(Sprite & sprite : *sprites)
    {
        deleteSprite(sprite);
    }
    delete sprites;
}
```

8.135 visual.hpp

Go to the documentation of this file.

```
#ifndef VISUAL
#define VISUAL

#include <memory>
#include <raylib.h>
#include <yaml-cpp/yaml.h>

#include <frame.hpp>

class Visual : public Frame
{
private:
    std::shared_ptr<Texture2D> m_texture;
    bool resizeable;
    static void deleteTexture2D(Texture2D*&);
protected:
    void fitFrame();
    void updateFrame(bool recursive = false) override;
public:
    Visual(Texture2D*, Frame*, Rectangle);
    Visual(Visual*);
    Visual(Visual*, Rectangle);
    Visual(Visual*, Frame*, Rectangle);
    ~Visual();

    void resize(fPoint);

    void draw();
};
using Sprite = std::vector<Visual*>;
void deleteSprite(Sprite*&);
void deleteSprites(std::vector<Sprite>*&);
#endif // VISUAL
```

8.136 src/window/include/window.hpp File Reference

```
#include <iostream>
#include <thread>
#include <chrono>
#include <vector>
#include <queue>
#include <stack>
#include <raylib.h>
#include <visual.hpp>
#include <action.hpp>
#include <container.hpp>
#include <object.hpp>
#include <interface.hpp>
#include <button.hpp>
```

Classes

- class Window
- class CloseAction

  manages the closing of the application
- class resizeAction

  manages the resizing of the window

8.137 window.hpp

Go to the documentation of this file.

```
#ifndef WINDOW_H
#define WINDOW_H

#include <iostream>
#include <thread>
#include <chrono>
#include <vector>
#include <queue>
#include <stack>

#include <raylib.h>

#include <visual.hpp>
#include <action.hpp>
#include <container.hpp>
#include <object.hpp>
#include <interface.hpp>
#include <button.hpp>


class Window
{
private:
    class InterfacePool
    {
    private:
        std::stack<Interface*> inf;
        std::map<std::string, Interface*> storage;
        void clearStack();
    public:
        InterfacePool();
        ~InterfacePool();
        void load(Interface*);
        void unload(Interface*);
        void clear();
        Interface* getInterface(std::string);

        void push(std::string);
        std::string pop();
        Interface* top();

        void draw();
        Action* react();
        Action* getRuntimeEvent();

    };
    class ActionPool
    {
    private:
        std::queue<Action*> pool;
        std::mutex mtx;
    public:
        ActionPool() = default;
        ~ActionPool();
        void push(Action* act);
        void push(PacketAction* act);
        Action* front();
        Action* pop();
        bool empty();
    };
    class WinContent
    {
    private:
        bool status;
        std::chrono::time_point<std::chrono::steady_clock> input_clock;
        std::chrono::time_point<std::chrono::steady_clock> runtime_clock;

        std::mutex status_mtx;
        std::mutex input_mtx;
        std::mutex runtime_mtx;
    public:
        std::chrono::duration<double> input_delay;
        std::chrono::duration<double> runtime_delay;
        float width;
        float height;
        Color background;
        std::string title;

        std::vector<std::thread> thread_pool;

        ~WinContent();

        void setStatus(bool);
        bool getStatus();

        void setInputClock2Now();
        void setRuntimeClock2Now();

        bool isInputDelayOver();
        bool isRuntimeDelayOver();

    };
    class UI
    {
    private:
        Frame* root_frame;
        InterfacePool* interface;

        std::mutex mtx;
        int reader;
        int writer;
        bool noRead;
        bool noWrite;
    protected:
        bool isReadable();
        bool isWritable();

        void reading();
        bool tryReading();
        void endReading();

        void writing();
        bool tryWriting();
        void endWriting();

        void DenyRead();
        void AllowRead();

        void DenyWrite();
        void AllowWrite();

    public:
        UI();
        ~UI();
        void draw();
        Action* react();
        Action* getRuntimeEvent();

        void setRootFrame(Frame*);
        Frame* getRootFrame();
        void resize(float, float);

        void setInterfacePool(InterfacePool*);

        void load(Interface*);
        void unload(Interface*);
        Interface* getInterface(std::string);

        void push(std::string);
        std::string pop();
        Interface* top();
    };

    friend class CloseAction;
    friend class resizeAction;

    WinContent Wcontent;
    UI UI;
    ActionPool immediate_user_pool, immediate_pool, request_pool, system_pool;

protected:
    void draw();
    void systemEvent();
    void getUserEvent();
    void getRuntimeEvent();
    void sound_effect();
    void immediateActing();
    void userActing();
    void requestActing();
    void systemActing();

    void initRaylib(YAML::Node node);
    void loadInterface(YAML::Node node);
    void loadGame(YAML::Node node);
public:
    Window();
    Window(std::string path);
    ~Window();

    bool isRun();
    bool isClose();
    void run();
};

class CloseAction : public Action
{
private:
    Window * win;
public:
    CloseAction(Window* win);
    ~CloseAction() = default;
    void execute();
};

class resizeAction : public Action
{
private:
    float w, h;
    Window* win;
public:
    resizeAction(Window* window, float w, float h);
    ~resizeAction() = default;
    void execute();
};
#endif
```

8.138 src/window/src/acting.cpp File Reference

```
#include <const/request.hpp>
#include <window.hpp>
```

8.139 acting.cpp

Go to the documentation of this file.

```
#include <const/request.hpp>
#include <window.hpp>

void Window::ActionPool::push(Action* action)
{
    std::lock_guard<std::mutex> lock(mtx);
    pool.push(action);
}

void Window::ActionPool::push(PacketAction* action)
{
    std::lock_guard<std::mutex> lock(mtx);
    std::vector<Action*> unpacked = action->unpack();
    delete action;

    for(Action*& a : unpacked)
    {
        pool.push(a);
    }

}

Window::ActionPool::~ActionPool()
{
    while(pop() != nullptr);
}

Action* Window::ActionPool::front()
{
    std::lock_guard<std::mutex> lock(mtx);
    return pool.front();
}

Action* Window::ActionPool::pop()
{
    std::lock_guard<std::mutex> lock(mtx);
    if(pool.empty()) return nullptr;
    Action* action = pool.front();
    pool.pop();
    return action;
}

bool Window::ActionPool::empty()
{
    std::lock_guard<std::mutex> lock(mtx);
    return pool.empty();
}


void Window::immediateActing()
{
    while(isRun())
    {
        Action* action = immediate_pool.pop();
        if(action == nullptr) continue;
        if(!isRun()) break;
        action->execute();
        delete action;
    }
}

void Window::userActing()
{
    while(isRun())
    {
        Action* action = immediate_user_pool.pop();
        if(action == nullptr) continue;
        if(!isRun()) break;
        action->execute();
        delete action;
    }
}

void Window::systemActing()
{
    {
        Action* action = system_pool.pop();
        if(action == nullptr) return;
        if(!isRun()) return;
        action->execute();
        delete action;
    }
}

void Window::requestActing()
{
    while(isRun())
    {
        Action* action = request_pool.pop();
        if(action == nullptr) continue;
        if(!isRun()) break;
        switch(action->isRequest())
        {
            case (REQUEST::ID::NONE):
                break;
            case (REQUEST::ID::INVALID):
                break;
            case (REQUEST::ID::CHANGE_INF):
                {
                    std::string id = action->getArgs().getInterfaceName();
                    UI.push(id);
                    break;
                }
            default:
                break;
        };

        delete action;
    }
}
```

8.140 src/window/src/action/close.cpp File Reference

```
#include <window.hpp>
```

8.141 close.cpp

Go to the documentation of this file.

```
#include <window.hpp>

CloseAction::CloseAction(Window* window)
{
    win = window;
}

void CloseAction::execute()
{
    win->Wcontent.setStatus(false);
}
```

8.142 src/window/src/action/resize.cpp File Reference

```
#include <window.hpp>
```

8.143 resize.cpp

Go to the documentation of this file.

```
#include <window.hpp>

resizeAction::resizeAction(Window* window, float x, float y)
{
    win = window;
    w = x;
    h = y;
}

void resizeAction::execute()
{
    win->UI.resize(w, h);
}


```

8.144 src/window/src/interface.cpp File Reference

```
#include <window.hpp>
```

8.145 interface.cpp

Go to the documentation of this file.

```
#include <window.hpp>

void Window::InterfacePool::clearStack()
{
    while(!inf.empty())
    {
        inf.pop();
    }
}

Window::InterfacePool::InterfacePool()
{
}
```

```

Window::InterfacePool::~InterfacePool()
{
    clear();
}

void Window::InterfacePool::load(Interface* i)
{
    storage[i->getName()] = i;
}

void Window::InterfacePool::unload(Interface* i)
{
    if(storage.find(i->getName()) != storage.end())
    {
        storage.erase(i->getName());
        delete i;
    }
}

void Window::InterfacePool::clear()
{
    for(auto i : storage)
    {
        delete i.second;
    }
    storage.clear();
    clearStack();
}

Interface* Window::InterfacePool::getInterface(std::string name)
{
    if(storage.find(name) != storage.end())
    {
        return storage[name];
    }
    return nullptr;
}

void Window::InterfacePool::push(std::string name)
{
    if(storage.find(name) != storage.end())
    {
        inf.push(storage[name]);
    }
}

std::string Window::InterfacePool::pop()
{
    if(!inf.empty())
    {
        std::string name = inf.top()->getName();
        inf.pop();
        return name;
    }
    return "";
}

Interface* Window::InterfacePool::top()
{
    if(!inf.empty())
    {
        return inf.top();
    }
    return nullptr;
}

void Window::InterfacePool::draw()
{
    if(!inf.empty())
    {
        inf.top()->draw();
    }
}

Action* Window::InterfacePool::react()
{
    if(!inf.empty())
    {
        return inf.top()->react();
    }
    return nullptr;
}

Action* Window::InterfacePool::getRuntimeEvent()
{
    if(!inf.empty())
    {
        return inf.top()->getRuntimeEvent();
    }
    return nullptr;
}
```

8.146 src/window/src/running.cpp File Reference

```
#include "raylib.h"
#include <window.hpp>
```

8.147 running.cpp

Go to the documentation of this file.

```
#include "raylib.h"
#include <window.hpp>

void Window::run() {
    // last_chrismas = now()
    Wcontent.setInputClock2Now();
    Wcontent.setRuntimeClock2Now();

    //Wcontent.thread_pool.push_back(std::thread(&Window::draw, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::getUserEvent, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::getRuntimeEvent, this));
    //Wcontent.thread_pool.push_back(std::thread(&Window::sound_effect, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::userActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::userActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::immediateActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::immediateActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::immediateActing, this));
    Wcontent.thread_pool.push_back(std::thread(&Window::requestActing, this));

    while (isRun())
    {
        draw();
        systemEvent();
        systemActing();
        //getUserEvent();
        //getRuntimeEvent();
        //sound_effect();
        //userActing();
        //immediateActing();
        std::this_thread::sleep_for(std::chrono::milliseconds(5));
    }
}

void Window::draw()
{
    {
        BeginDrawing();
        UI.draw();
        EndDrawing();
    }

}

void Window::systemEvent()
{
    {
        // alt + F4 to exit
        if (IsKeyDown(KEY_LEFT_ALT) && IsKeyDown(KEY_F4))
        {
            system_pool.push(new CloseAction(this));
        }
        if (WindowShouldClose())
        {
            system_pool.push(new CloseAction(this));
        }

        if (IsWindowResized() && !IsWindowFullscreen())
        {
            int width = GetScreenWidth();
            int height = GetScreenHeight();
            system_pool.push(new resizeAction(this, width, height));
        }
    }
}

void Window::getUserEvent()
{
    while(isRun())
    {
        if(!Wcontent.isInputDelayOver())
        {
            std::this_thread::sleep_for(std::chrono::milliseconds(10));
            continue;
        }

        Action* action = UI.react();
        if(action != nullptr)
        {
            if(action->isPackage())
            {
                for(auto act : action->unpack())
                {
                    if(act->isRequest())
                        request_pool.push(act);
                    else
                        immediate_user_pool.push(act);
                }
            }
            else if(!action->isRequest())
                immediate_user_pool.push(action);
        }

        Wcontent.setInputClock2Now();
    }
}

void Window::getRuntimeEvent()
{
    while(isRun())
    {

        if(!Wcontent.isRuntimeDelayOver())
        {
            std::this_thread::sleep_for(std::chrono::milliseconds(10));
            continue;
        }
        Action* action = UI.getRuntimeEvent();
        if(action != nullptr)
        {
            if(action->isPackage())
            {
                for(auto act : action->unpack())
                {
                    if(act->isRequest())
                        request_pool.push(act);
                    else
                        immediate_user_pool.push(act);
                }
            }
            else if (!action->isRequest())
                immediate_pool.push(action);
        }
        Wcontent.setRuntimeClock2Now();
    }
}
void Window::sound_effect()
{
    // do nothing
}

bool Window::isRun()
{
    return Wcontent.getStatus();
}

bool Window::isClose()
{
    return !Wcontent.getStatus();
}
```

8.148 src/window/src/UI.cpp File Reference

```
#include "action.hpp"
#include <window.hpp>
#include <mutex>
```

8.149 UI.cpp

Go to the documentation of this file.

```
#include "action.hpp"
#include <window.hpp>
#include <mutex>

Window::UI::UI()
{
    root_frame = nullptr;
    interface = nullptr;
}

Window::UI::~UI()
{
    if(root_frame != nullptr) delete root_frame;
    if(interface != nullptr) delete interface;
}

void Window::UI::draw()
{
    if(!tryReading()) return ;
    interface->draw();
    endReading();
}

Action* Window::UI::react()
{
    if(!tryReading()) return nullptr;
    Action* act = interface->react();
    endReading();

    return act;
}

Action* Window::UI::getRuntimeEvent()
{
    if(!tryReading()) return nullptr;
    Action* act = interface->getRuntimeEvent();
    endReading();

    return act;
}

void Window::UI::setRootFrame(Frame* frame)
{
    if(root_frame != nullptr)
        delete root_frame;
    root_frame = frame;
}

Frame* Window::UI::getRootFrame()
{
    return root_frame;
}

void Window::UI::resize(float width, float height)
{
    writing();
    root_frame->resize(width, height);
    endWriting();
}

void Window::UI::setInterfacePool(InterfacePool* inter)
{
    if(interface != nullptr)
        delete interface;
    interface = inter;
}

```

```
void Window::UI::load(Interface* inf)
{
    interface->load(inf);
}

void Window::UI::unload(Interface* inf)
{
    interface->unload(inf);
}

Interface* Window::UI::getInterface(std::string s)
{
    reading();
    Interface* f = interface->getInterface(s);
    endReading();
    return f;
}


void Window::UI::push(std::string s)
{
    writing();
    interface->push(s);
    endWriting();
}

std::string Window::UI::pop()
{
    writing();
    std::string f =  interface->pop();
    endWriting();
    return f;
}

Interface* Window::UI::top()
{
    reading();
    Interface* f = interface->top();
    endReading();
    return f;
}

bool Window::UI::isReadable()
{
    std::lock_guard<std::mutex> lock(mtx);
    return !noRead;
}

bool Window::UI::isWritable()
{
    std::lock_guard<std::mutex> lock(mtx);
    return !noWrite;
}

void Window::UI::reading()
{
    do
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(5));
        std::lock_guard<std::mutex> lock(mtx);
        if(noRead) return ;
        reader++;
        return ;
    }while(true);
}

bool Window::UI::tryReading()
{
    std::lock_guard<std::mutex> lock(mtx);
    if(noRead) return false;
    reader++;

    return true;
}

void Window::UI::endReading()
{
    std::lock_guard<std::mutex> lock(mtx);
    reader--;
}

void Window::UI::writing()
{
    do
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(5));
        std::lock_guard<std::mutex> lock(mtx);
        if(noWrite) return ;
        writer++;
        noRead = true;
        return;
    }while(true);
}

bool Window::UI::tryWriting()
{
    std::lock_guard<std::mutex> lock(mtx);
    if(noWrite) return false;
    writer++;
    noRead = true;
    return true;
}

void Window::UI::endWriting()
{
    std::lock_guard<std::mutex> lock(mtx);
    writer--;
    if(writer == 0) noRead = false;
}

void Window::UI::DenyRead()
{
    std::lock_guard<std::mutex> lock(mtx);
    noRead = true;
}

void Window::UI::AllowRead()
{
    std::lock_guard<std::mutex> lock(mtx);
    noRead = false;
}

void Window::UI::DenyWrite()
{
    std::lock_guard<std::mutex> lock(mtx);
    noWrite = true;
}

void Window::UI::AllowWrite()
{
    std::lock_guard<std::mutex> lock(mtx);
    noWrite = false;
}


```

8.150 src/window/src/wincontent.cpp File Reference

```
#include <window.hpp>
```

8.151 wincontent.cpp

Go to the documentation of this file.

```
#include <window.hpp>

Window::WinContent::~WinContent()
{
    for(std::thread &i : thread_pool)
    {
        if(i.joinable())
            i.join();
    }
}


void Window::WinContent::setStatus(bool b)
{
    std::lock_guard<std::mutex> lock(status_mtx);
    status = b;
}

```

```
bool Window::WinContent::getStatus()
{
    std::lock_guard<std::mutex> lock(status_mtx);
    return status;
}

void Window::WinContent::setInputClock2Now()
{
    std::lock_guard<std::mutex> lock(input_mtx);
    input_clock = std::chrono::steady_clock::now();
}

void Window::WinContent::setRuntimeClock2Now()
{
    std::lock_guard<std::mutex> lock(runtime_mtx);
    runtime_clock = std::chrono::steady_clock::time_point();
}

bool Window::WinContent::isInputDelayOver()
{
    std::lock_guard<std::mutex> lock(input_mtx);
    return std::chrono::steady_clock::now() - input_clock > input_delay;
}

bool Window::WinContent::isRuntimeDelayOver()
{
    std::lock_guard<std::mutex> lock(runtime_mtx);
    return std::chrono::steady_clock::now() - runtime_clock > runtime_delay;
}

```