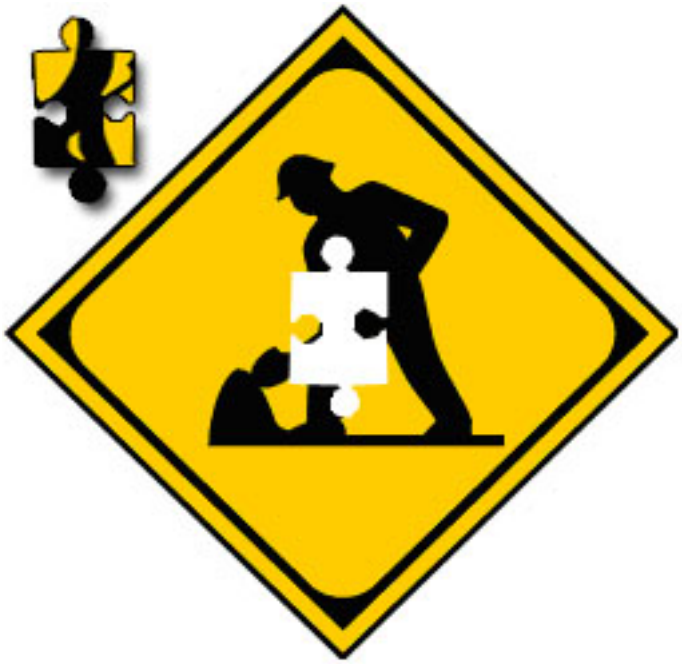
UC - Developer Guide



Under Construction

Stefan Hildenbrand
Florian Keusch
Markus Neidhart
Beat Steiger

Inhaltsverzeichnis

1. Einführung **5**
- 1.1. Zweck des Dokuments . . . 5
- 1.2. Zweck des Softwareprodukts . . . 5
- 1.3. Erläuterungen zu Begriffen . . . 5
- 1.4. Referenzen . . . 6
- 1.5. Entwickler . . . 6
- 1.6. Übersicht über das restliche Dokument . . . 6

2. Installation **7**
- 2.1. Eiffel . . . 7
- 2.2. ESDL . . . 7
- 2.3. Umgebungsvariablen . . . 7
- 2.4. diverse hacks . . . 8
  - 2.4.1. Mixer für der Sound . . . 8
  - 2.4.2. Alpha Kanal für Bilder . . . 8

3. Implementation **10**
- 3.1. Funktionsbeschreibung . . . 10
  - 3.1.1. Mausevents . . . 10
  - 3.1.2. Keyevents . . . 10
  - 3.1.3. Buttons (play window) . . . 11
  - 3.1.4. New Game . . . 11
  - 3.1.5. Resume Game . . . 11
  - 3.1.6. Save Game . . . 11
  - 3.1.7. Load Game . . . 12
- 3.2. Files and Clusters . . . 12
- 3.3. Spezielle Klassen . . . 13
  - 3.3.1. TEXT_FIELD.e . . . 13
  - 3.3.2. BUTTON.e . . . 13
  - 3.3.3. MUSIC_PLAYER.e . . . 13
  - 3.3.4. RANDOM_NUMBER_GENERATOR.e . . . 14
  - 3.3.5. UC_SHARED_ITEMS.e . . . 14
- 3.4. Patterns . . . 14
  - 3.4.1. mvc-pattern . . . 14
  - 3.4.2. observer-pattern . . . 15
  - 3.4.3. command-pattern . . . 15
  - 3.4.4. abstract-factory-pattern . . . 15
- 3.5. Umgehen mit verschiedenen Szenen . . . 15
- 3.6. Klassen . . . 16
  - 3.6.1. cutter/QUAD_CUTTER.e . . . 16
  - 3.6.2. cutter/SHARED_QUAD_CUTTER.e . . . 18
  - 3.6.3. gui_elements/BUTTON.e . . . 18
  - 3.6.4. gui_elements/TEXT_FIELD.e . . . 21

3.6.5. menus/CREDITS.e . . . 24
3.6.6. menus/LOAD_GAME.e . . . 25
3.6.7. menus/MENU.e . . . 27
3.6.8. menus/NEW_GAME.e . . . 28
3.6.9. menus/PREDEFINED_GAME.e . . . 30
3.6.10. menus/SAVE_GAME.e . . . 30
3.6.11. menus/SELECT_GAME.e . . . 31
3.6.12. menus/UC_INTERFACE.e . . . 32
3.6.13. model_view/UC_PUZZLE.e . . . 32
3.6.14. model_view/UC_PUZZLE_PART.e . . . 34
3.6.15. model_view/UC_PUZZLE_PART_VIEW.e . . . 36
3.6.16. model_view/UC_PUZZLE_VIEW.e . . . 38
3.6.17. sound/ESDL_AUDIO_CONSTANTS.e . . . 39
3.6.18. sound/ESDL_MIXER.e . . . 40
3.6.19. sound/ESDL_MUSIC.e . . . 43
3.6.20. sound/MUSIC_PLAYER.e . . . 44
3.6.21. workspace/UC_FROM_SPACE . . . 47
3.6.22. workspace/UC_TO_SPACE.e . . . 48
3.6.23. workspace/UC_WORKSPACE.e . . . 49
3.6.24. FINISH_SCREEN.e . . . 51
3.6.25. PUZZLE_START_WINDOW.e . . . 51
3.6.26. RANDOM_NUMBER_GENERATOR.e . . . 56
3.6.27. UC.e . . . 58
3.6.28. UC_LOAD_PUZZLE.e . . . 58
3.6.29. UC_SAVE_PUZZLE.e . . . 59
3.6.30. UC_SHARED_ITEMS.e . . . 60
3.7. BON-Diagram . . . 63

A. Appendix **64**
A.1. Vergleich zum SRS . . . 64
A.1.1. erreichte Punkte . . . 64
A.1.2. nicht erreichte Punkte . . . 65
A.1.3. Fazit . . . 65
A.2. Entwicklung des Projekts . . . 66
A.2.1. Entwicklungsumgebung . . . 66
A.2.2. Start . . . 66
A.2.3. Arbeitsverteilung . . . 66
A.2.4. Probleme . . . 66
A.3. Testing . . . 67
A.3.1. Bekannte Bugs . . . 67
A.4. Weiterentwicklungsmöglichkeiten . . . 68
A.4.1. Sound . . . 68
A.4.2. Modus . . . 68
A.4.3. Fancy Puzzle Part . . . 68
A.4.4. Join von mehreren Puzzle Teilen . . . 68
A.4.5. Scrollfunktionen . . . 68
A.4.6. Zoom . . . 68
A.4.7. Filter / Search . . . 68
A.5. Wiederverwendbarkeit . . . 69
A.5.1. Music-Player . . . 69
A.5.2. Gui-Elemente . . . 69
A.5.3. Zufalls-Generator . . . 69
A.5.4. Mausunterstützung im Menu . . . 69

A.5.5. Behandlung von Key- und anderen Events 69
A.5.6. Weiteres 69
A.6. Special Thanks to 69

1. Einführung

1.1. Zweck des Dokuments

Dieses Dokument ist das Entwickler Handbuch zum Spiel "Under Construction" und richtet sich hauptsächlich an Entwickler und Weiterentwickler dieses Spiels. Durch dieses Dokument soll der Leser informiert werden über den Aufbau unseres Projekts, über verwendete Patterns, über Wiederverwendbarkeit einzelner Klassen, und auch Details des Sourcecodes werden aufgelistet.

1.2. Zweck des Softwareprodukts

Unsere Software ist ein Puzzlepiel und heisst "Under Construction", kurz UC. Das Spiel wird:

1. ein geladenes Bild in kleine Teile zerlegen.
2. dem Spieler die Möglichkeit geben, selbst ein beliebiges Bild zu laden.
3. dem Spieler die Möglichkeit geben, die Anzahl der Teile und somit den Schwierigkeitsgrad zu bestimmen.
4. einen ETH-Modus bieten, der ein Bild der ETH lädt und es in 150 Teile zerlegt.
5. 3 Arbeitsflächen zur Verfügung stellen, auf denen der Spieler seine Teile sortieren kann.
6. Sound kann abgespielt werden.

Das Ziel des Spiels ist es, alle Teile so aneinanderzureihen, dass wieder das ursprüngliche Bild entsteht, also das Puzzle zu lösen.

1.3. Erläuterungen zu Begriffen

Spielfläche Teil des Displays, der als Ablage für diejenigen Teile dient, die das fertige Ursprungsbild ergeben.

Arbeitsfläche (workspaces) Teil des Displays, um noch nicht zur Spielfläche hinzugefügte Teile zu sortieren.

play window (grahisch) Ist das gesamte Fenster, in dem der Spieler seine Puzzleteile zusammensetzt oder den Player steuern kann. (dazu gehört also Spielfläche und Arbeitsfläche)

Puzzle Start Window PSW Ist jene Klasse, die das ganze PuzzleSpiel initialisiert und managt.

1.4. Referenzen

- Eiffel, http://www.eiffel.com
- ESDL, http://eiffelsdl.sourceforge.net
- http://se.inf.ethz.ch/download/games/developer/
- ESDL, http://eiffelsdl.sourceforge.net
- EWG: Eiffel wrapper generator: http://ewg.sf.net/
- Gobo: http://www.gobosoft.com/eiffel/gobo/
- SDL: http://libsdl.org/
- Chair of Software Engineering: http://se.inf.ethz.ch/
- VIS Forum: http://forum.vis.ethz.ch

1.5. Entwickler

Die Entwickler dieses Projekts sind:

Stefan Hildenbrand	stefanhi@student.ethz.ch
Florian Keusch	fkeusch@student.ethz.ch
Beat Steiger	steigebe@student.ethz.ch
Markus Neidhart	nemarkus@student.ethz.ch

Alle sind zu erreichen unter: uc@floesen.ch

Dieses Projekt wurde entwickelt im Rahmen der Vorlesung Software Architecture, die an der ETH Zürich im Sommersemester 05 gehalten wurde.

1.6. Übersicht über das restliche Dokument

Im restlichen Teil dieses Dokuments wird auf die Installation von allen benötigten Teilen eingegangen. Die Funktionen werden erklärt und Abhängigkeiten aufgelistet. Weiterhin werden die features und requirements aller unserer Klassen aufgelistet. Deren Abhängigkeiten werden im BON-Diagram graphisch dargestellt.

2. Installation

2.1. Eiffel

Um das Spiel zu entwickeln wir das Eiffel Studio (www.eiffel.com) benötigt. Für die Entwicklung dieses Projekts wurde vorwiegend Eiffel Version 5.5 unter Windows verwendet. Für die Entwicklung haben wir den freien bcb compiler verwendet (bei der Installation von Eiffel kann ausgewählt werden). Unter Linux wurde das Spiel nicht getestet. Es ist jedoch möglich dieselbe Funktionalität unter Linux zu erreiche, wenn erst mal das ganze ESDL mitsamt den benötigten libraries installiert wurde. (Dazu benötigt man die Linux Version von ESDL, die cvs sourcen von ewg und gobo, ev auch noch sdl libraries. Zur Installation unter Linux beachte das file developer.txt (zu ESDL) und diverse Foren im Internet)

2.2. ESDL

Im Spiel benötigt wurde die Version 0.6.0 von EDSL. Zu finden unter:
http://se.inf.ethz.ch/download/games/developer/
Weiterhin wird gobo und ewg benötigt. (die aber unter Windows automatisch installiert werden (ESDL). Ebenso sollten die Umgebungsvariablen automatisch gesetzt werden.)

2.3. Umgebungsvariablen

Damit alles richtig funktioniert, sollten etwa folgende Umgebungsvariablen gesetzt sein:

- ESDL=$ESDL
- EWG=C:$ESDL/dependency/ewg
- GOBO=C:$ESDL/dependency/gobo
- GOBO_CC=bcb
- GOBO_EIFFEL=ise
- ISE_C_COMPILER=bcb
- ISE_EIFFEL=$EIFFEL
- Path=$EDSL/dependency/sdl/lib
- Path=$ESDL/dependency/ewg/bin
- Path=$ESDL/dependency/gobo/bin
- SDL=$ESDL/dependency/sdl
- SDL_HEADER=$ESDL/dependency/sdl/include

Wobei $ESDL der Pfad ist, der auf das ESDL directory zeigt. $EIFFEL ist der Pfad, in dem Eiffel installiert wurde.

2.4. diverse hacks

2.4.1. Mixer für der Sound

Um den Mixer zu includen muss man das gesamte ESDL neu zu kompilieren. Zuerst noch die Umgebungsvariablen überprüfen, dann folgende Schritte durchführen (Achtung, hier wurde der compiler bcb verwendet, sonst anpassen):

1. neuste ESDL Version downloaden (hier wurde: esdl_bcb_0.6.0.exe verwendet).
2. neue esdl version installieren falls nötig
3. in der Datei $ESDL/library/manual_wrapper/c/include/esdl.h den auskommentierten SDL_mixer.h einbinden.
4. neu kompilieren (im ordner $ESDL/library)
   a) *geant clobber*
   b) *geant clean* (→ alles wird gelöscht)
   c) *geant install* (→ kompilierung wird vorbereitet)
   d) *geant c_build_library_ise* (oder so ähnlich, je nach compiler) (→ wird neu kompiliert)

   Durch eingeben des Befehls *geant* werden alle möglichen Befehle aufgelistet.
5. alles sollte nun funktionieren ;-)

Damit man keine c-compilation Fehler kriegt, muss eventuell zusätzlich im Eiffelstudio bei project → project settings unter dem register externals das object-file *$(SDL)/lib/SDL_mixer.lib* hinzugefügt werden

2.4.2. Alpha Kanal für Bilder

Um 32bit Bilder mit Alphakanal laden zu können, müssen Bugs in den zwei Klassen ESDL_BITMAP_FACTORY und ESDL_SURFACE korrigiert werden. Danach werden TGA und PNG Bilder mit Transparenz korrekt dargestellt (In BMPs wird der Alphakanal nicht erkannt/gelesen).
(siehe auch http://forum.vis.ethz.ch/thread.php?threadid=6135)

Aenderungen sind mit ***** markiert:

```
Klasse ESDL_BITMAP_FACTORY
Feature create_bitmap_from_image


if temp_image_pointer /= default_pointer then
    image_pointer := sdl_display_format_alpha_external (temp_image_pointer)
                     *******************************
    sdl_free_surface_external (temp_image_pointer)
    if image_pointer /= default_pointer then
```

```
      create last_bitmap.make_from_pointer (image_pointer)
    end
end
```

```
Klasse ESDL_SURFACE
Feature gl_texture
```

```
if
    texture_modified
then
    saved_flags := flags & (Esdl_srcalpha | Esdl_rleaccelok)
    saved_alpha := pixel_format.alpha
    if
        saved_flags & Esdl_srcalpha = Esdl_srcalpha
    then
        error := sdl_set_alpha_external (current.item, 0, 0)
    end                                  ************
        -- blit to surface with correct (power of 2) dimensions
    surf_tex_.blit_surface (current, 0, 0)
        -- restore the alpha blending attributes
    if
        saved_flags & Esdl_srcalpha = Esdl_srcalpha
    then
        error := sdl_set_alpha_external (current.item, saved_flags, saved_alpha)
    end                                  ************
```

```
Klasse ESDL_SURFACE
Feature gl_texture_mipmap
```

Gleiche Änderung wie in Feature gl_texture (gleiche 2 Stellen, jeweils $current durch current.item ersetzen)

3. Implementation

3.1. Funktionsbeschreibung

3.1.1. Mausevents

- Puzzleteile können mit der linken Maustaste bewegt werden. Der Spieler zieht ein passendes Teil vom Workspace in den Spielraum. Wenn dieses Teil passt, dann schnappt es ein, sonst spickt es zurück in den Workspace. Befindet sich das Teilchen noch immer über einem Workspace, wird es an der entsprechenden Stelle einsortiert. Ebenfalls reagieren die Buttons auf ein Teilchen, das über ihnen 'fallen gelassen' wird.
- Mit der rechten Maustaste wird ein Puzzleteil um 90 Grad gedreht. Durch 4 mal klicken erhält man also wieder das ursprüngliche Puzzleteilchen.
- durch Klicken auf einen Button wird das zugehörige event Ausgelöst und der Button blinkt kurz auf.
- Wenn ein Button mit der Maus überfahren wird, so ändert sich dessen Hintergrundfarbe.
- Wenn auf ein Textfeld geklickt wird, so erscheint dort ein blinkender Cursor. Dann ist das Textfeld bereit für die Eingabe. Durch weggklicken auf irgendeine andere Fläche wird der Focus verschoben.

3.1.2. Keyevents

- Mit der Taste ESC wird das jeweilige Fenster verlassen. Ist der Spieler im Hauptmenu wird damit das Spiel beendet.
- Mit der Taste p (pause/resume) wird der Musikplayer angehalten, respektive er wird fortgesetzt.
- Die Taste s (stopp) stoppt den Musikplayer.
- Durch Druck der Taste q wird das Program sofort verlassen.
- Die Taste b (back) spielt das vorherige Lied in der Playliste ab.
- Die Taste n (next) spielt das nächste Lied ab.
- Die Taste down bewirkt, dass sich im Menu die Auswahl um eins nach unten bewegt. Im play window kann damit die Lautstärke erhöht werden.
- Die Taste up bewirkt, dass sich im Menu die Auswahl um eins nach oben bewegt. Im play window kann damit die Lautstärke gesenkt werden.
- Die Taste m (mute) schaltet den Ton aus. Der Player läuft aber im Hintergrund weiter. Durch ein 2tes mal drücken schlatet der Ton wieder ein.
- Die Taste t (title) schaltet die Anzeige des Titels aus / ein.

- Die Taste c (cheat) setzt ein Puzzle Teilchen an die richtige Stelle.
- Die Taste i (image) zeigt das ganze Bild (Lösung) für 3 Sekunden an.
- Die Taste r (reset) formatiert die workspaces neu.
- Mit den Tasten 0 bis 2 können die 3 workspaces angewählt werden.
- Die Taste d (delete) löscht das aktuelle angeklickte Puzzle Teil aus dem workspace. Falls keines angeklickt ist werden alle Puzzle Teile aus dem workspace gelöscht.
- Enter aktiviert die aktuelle Menuauswahl im Menu (Wenn nur eine Auswahl, dann wird die ausgelöst).

3.1.3. Buttons (play window)

Im play window befinden sich diverse Buttons mit den folgenden Funktionen:

- workspace Buttons repräsentieren den jeweiligen Workspace. Durch einen Klick darauf wird derjenige Workspace angezeigt. Schiebt man ein Puzzle Teil über den Button so wird diese Teil in diesen Workspace verschoben.
- Player Buttons dienen zur Steuerung des Musik Players.
- Der Cheat Button (Zauberstab) setzt ein Puzzle Teilchen an die richtige Stelle im Puzzle.
- Der Bild Button (Auge) zeigt das Bild (die fertige Lösung) für 3 Sekunden an.

3.1.4. New Game

Hier kann der Spieler ein neues Puzzle erstellen. Es gibt 2 verschiedene Varianten. Der Spieler kann ein vordefiniertes Puzzle aufwählen, oder kann selber das Bild und Anzahl Teile bestimmen: In einem Textfeld werden Namen des Bildes angegeben und Anzahl Teile. (z.b. *hello_world.gif*, das Bild muss sich im Ordner image befinden). Alle Bildformate können ausgewählt werden, die von der Klasse ESDL_BITMAP_FACTORY unterstützt werden. Die Informationen, die dort angegeben werden, werden als Strings an das PUZZLE_START_WINDOW weitergeleitet. Dort wird das ganze Puzzle kreiert und initialisiert. (alle .ogg files im Ordner sound werden per default abgespielt)

3.1.5. Resume Game

Wenn der Spieler aus dem play window gegangen ist, kann er mit dieser Auswahl das angefangene Spiel fortsetzen. Dabei wird die vorher kreierte Scene einfach durch das Menu weitergereicht, um den Spielstand im Hintergrund zu sichern (siehe: Umgehen mit verschiedenen Szenen).

3.1.6. Save Game

Hier hat der User die Möglichkeit ein bereits angefangenes Puzzle zu speichern, um ein andermal weiterspielen zu können. Implementiert wurde dies etwa so:
Man das PSW dem UC_INTERFACE übergeben (siehe: Umgehen mit verschiedenen Szenen). Dann wird dieses jedoch bei Klick auf Save_game weitergereicht an die Klasse Save_game, wo das Puzzlebild und die Anzahl Puzzleteile in ein Textfile geschrieben werden (aus PSW extrahiert). Dies reicht, da ein cutter ein bestimmtes Bild mit einer bestimmten Anzahl Teilen immer gleich cuttet, somit sind auch die puzzle_parts immer die gleichen. Was aber zu speichern ist: welche puzzle_part_views in

welchen Workspaces liegen und welche Eigenschaften sie haben (locked, orientation, x, y, zugehöriges puzzle_part). Deshalb wird jeder Workspace via Loop durchlaufen (ein Workspace ist dabei ein ESDL_DRAWABLE_CONTAINER, also eine Liste mit drawables) und von jedem puzzle_part_view diese Informationen gespeichert.

3.1.7. Load Game

Hier hat der User die Möglichkeit ein bereits gespeichertes Puzzle zu laden und weiterzuspielen. Das Laden wurde folgendermassen implementiert:
Zuerst wird vom Textfile Bildname und Anzahl Teile gelesen und via cutter gebastelt. Dann wird jeder workspace wieder zusammengebaut, indem er erst mal komplett kreiert wird von einem puzzle (cutter.last_puzzle). Dann sind aber alle parts als puzzle_part_views drinn, was nicht sein muss. Also wird wipe_out aufgerufen und alle puzzle_part_views wieder rausgeworfen (nur Parameter wie Breite und Höhe,... sollen gespeichert bleiben). Dann werden alle puzzle_part_views im jeweiligen Workspace ausgelesen und kreiert, anschliessend in den workspace gelegt. Sind alle workspaces soweit gemacht, dann wird ein PSW erstellt. Dieses darf aber ähnlich wie bei Resume Game jetzt nicht einfach inizialize_scene aufrufen, da sonst alle workspaces wieder überschrieben werden. Stattdessen wird eine Funktion initialize_loaded aufgerufen, welche nur jene funktionen aufruft, die nicht load-spezifisch sind (also z.B. die Menge puzzle_parts_in_row für alle workspaces richtig setzen oder image und Anzahl Teile vom cutter holen, usw). Danach hat existiert ein korrektes PSW, welches dem entspricht, das man abgespeichert hat. Nun wird dieses als last_scene von UC_INTERFACE gespeichert und der Rest ist identisch zum resume_game (siehe: Umgehen mit verschiedenen Szenen).

3.2. Files and Clusters

release eine lauffähige kompilierte Version des Projekts mit allen benötigten Daten und Bildern

sound Dies ist der default Ordner, in dem das Programm nach Musik sucht. Ebenfalls befinden sich hier alle Klassen, die etwas mit der Ausgabe des Soundes von UC zu tun haben.

sound\applaus das File applaus im Ordner sound wird abgespielt, wenn das finish_window ausgelöst wird, also das Puzzle fertig ist.

image alle benötigten Bilder und Icons, sowie die verwendeten Schriften.

image\uc_logo.png Bild, das im Menu angezeigt wird

image\icon.png das Applikationsicon das oben links angezeigt wird

cutter Klassen, die für die Zerlegung der Bilder zuständig sind

gui_elements Klassen, die für die Darstellung des gui benötigt werden

model_view Klassen die das Puzzle bauen und für die Darstellung der Puzzleteilchen zuständig sind.

workspace Klassen, die für einen workspace benötigt werden

menus Klassen, welche die verschiedenen Menus des Spiels erstellen

ise.ace Das File ise.ace wird benötigt um in Eiffel das ganze zu kompilieren. Dieses .ace-File ist für die Windows-Plattform. Um ein .ace-File für eine andere Version zu erhalten, benutze man *geant install*. Die für diesen Befehl notwendigen Files befinden sich ebenfalls im Ordner. Die GOBO-Tools wurden mit ESDL installiert.

3.3. Spezielle Klassen

3.3.1. TEXT_FIELD.e

Diese Klasse ist ein Gui Element und stellt ein Eingabefeld zur Verfügung. Klickt man auf dieses Textfeld, so beginnt ein Curser zu blinken. Nun kann der User in dieses Textfeld einen beliebigen String eingeben. Ein Textfeld kann benutzt werden, um die Kommunikation zwischen Spieler und Program zu ermöglichen. Der im Textfeld angegebene String kann im Program weiterverwendet werden (z.b. als Pfadangabe für das Bild das zu laden ist).

Das Textfeld kann für andere Applikation beliebig verwendet und eingebunden werden. Viele features wie: *set_font, set_background_color, set_cursor_color, set_border_color, set_border_width, set_width, set_curser_blinkrate*, ... bieten die Möglichkeit diese Textfeld speziell anzupassen und in eine grafische (ESDL) Applikation einzubinden.

3.3.2. BUTTON.e

Die Klasse BUTTON.e stellt ein Gui Element Button zur Verfügung. Wenn der Spieler auf so einen Button klickt, so kann dadurch ein bestimmtes Ereignis ausgelöst werden. Wenn der Spieler den Button mit der Maus überfährt, so ändert sich seine Farbe. Seine Farbe ändert sich abermals, wenn auf den Button geklickt wird. Der Button kann ebenfalls 'aktiviert' werden, dann er wieder eine andere Farbe.

Diese Klasse kann für andere Projekte angepasst und weiterverwendet werden. Mit features wie: *set_width, set_height, set_vertical_margin, set_horizontal_margin, set_name, set_normal_color, set_prelight_color, set_highlight_color, set_border_color, set_icon, set_font, set_corners_roundes, set_border_width*, und weitere, hat man viele Möglichkeiten einen Button nach belieben anzupassen und in seine graphische Applikation (ESDL) einzubinden.

In unserem Projekt wurde diese Klasse verwendet um den Music Player zusätzlich mit der Maus steuern zu können. Ebenfalls haben wir sie gebraucht um die Steuerung der verschiedenen Workspaces zur Verfügung zu stellen.

Um die Buttons gut zu animieren ist es von Vorteil in deren Icons transparenten Hintergrund zu haben.

3.3.3. MUSIC_PLAYER.e

Diese Klasse stellt einen kompletten Musik Player zur Verfügung. Der Musik Player bietet folgende Funktionen: (nachdem er initialisiert wurde: make liest das default sound directory aus und initialisiert den musictrack / nach dem auslesen des sound directories existiert ein container sound_container, in dem alle abspielbaren Files drin sind)

- *play* öffnet den aktuellen song und spielt ihn ab
- *stop* stoppt den player
- *pause* pausiert den player oder beginnt wieder an der pausierten Stelle zu spielen (falls schon pausiert)

- *next* spielt den nächsten song ab
- *previous* spielt den vorherigen song ab
- *set_volume* setzt die Lautstärke
- *turn_up* erhöht die Lautstärke um 10
- *turn_down* senkt die Lautstärke um 10
- *set_sound_dir (a_string: STRING)* setzt ein neues sound directory und liest dieses gleich aus
- *next_song* gibt den String des nächsten songs zurück
- *previous_song* gibt den String des letzten songs zurück
- *set_event_loop (an_ev_loop: ESDL_EVENT_LOOP)* setzt das Ereiegnis check next auf den event_loop. Dies wird benötigt, um zu überprüfen, ob das Lied fertig ist. Falls ja, wird automatisch das nächste abgespielt.
- *mute* setzt die Lautstärke auf 0, das Lied spielt aber weiter.
- diverse status checks (is_playing, is_stopped, is_paused, volume, ...)
- *song_title* gibt den gesamten String zurück des spielenden Liedes.
- *song_title_cut* gibt den String des Liedes zurück ohne Endung und sound directory
- ...

3.3.4. RANDOM_NUMBER_GENERATOR.e

Diese Klasse stellt einige random Funktionen zur Verfügung. Sie wird mit der Systemzeit initialisiert (seed) und gibt bei Anforderung immer den nächsten Zufallswert im verlangten Format (Integer, Integer (mod n), Real, Double, Boolean) zurück.

3.3.5. UC_SHARED_ITEMS.e

Viele Klassen unsers Projekts erben von UC_SHARED_ITEMS.e. Diese Klasse wurde entwickelt, um auf immer wieder gebrauchte Objekte Zugriff zu haben und damit diese nicht wieder erstellt werden müssen. Um die Klasse zu instanzieren werden viele Variablen mit *once* einmal kreiert.

3.4. Patterns

Bei der Entwicklung unseres Projekt verwendeten wir verschiedene Patterns:

3.4.1. mvc-pattern

Das Model-View-Controller-Pattern findet sich in der Implementation der Puzzle-Teilchen. Es gibt einerseits die Klasse puzzle_part und andererseits die Klasse puzzle_part_view.

Die puzzle_parts stellen das Modell des Puzzle-Teilchens zur Verfügung. Insbesondere bieten sie die rotierten Ansichten des Teilchens an und kennen ihren Platz innerhalb des Puzzles.

Die puzzle_part_views dagegen wissen, wo auf dem Screen sie dargestellt werden und welche Rotation gerade aktuell ist. Anfragen über die korrekte Position werden an das puzzle_part weitergereicht und um die Abfrage der Korrekten Rotation ergänzt.

Es kann mehrere views auf das selbe Teilchen geben, dadurch kann auch Speicherplatz gespart werden.

Das Pattern konnte nicht konsequent umgesetzt werden, da die views die *draw*-Funktion ebenfalls an das Teilchen weitergeben wird. Somit in diesem Moment die Position auch dem Teilchen selbst bekannt sein muss, damit es sich richtig auf eine Oberfläche zeichnen kann. Ebenfalls bei der Abfrage der korrekten Position muss dem Teilchen seine Position auf der Oberfläche bekannt sein.

Trotzdem halten wir das Pattern für gerechtfertigt, insbesondere, da ein Teilchen ja in mehreren Workspaces angezeigt werden kann und dafür nur das view kopiert werden muss und nicht das Teilchen selbst.

3.4.2. observer-pattern

Das observer-Pattern findet durch das ganze Spiel Anwendung. Die von ESDL zur Verfügung gestellten Events werden genutzt um entsprechende Handlungen auszuführen.

3.4.3. command-pattern

Die Behandlung des key-down-events kann als Anwendung des command-Patterns betrachtet werden. Der User fordert eine Aktion an und der erste Agent versucht herauszufinden, welchem weiteren Agent er diese Anforderung weitergeben kann.

3.4.4. abstract-factory-pattern

Die cutter-Klasse ist als eine Factory implementiert. Man benutzt Singleton-Access auf diese Factory und erstellt damit seine Puzzle-Teilchen. Diese Implementation sollte es ermöglichen ohne Probleme eine weitere Art von Puzzle-Teilchen (z.B. Teilchen mit anderem Rand) zu erzeugen, die genau gleich verwendet werden können.

3.5. Umgehen mit verschiedenen Szenen

Das Problem beim Resume war ja in erster Linie, das vorhandene PUZZLE_START_WINDOW - wenn man zurück ins menu geht - irgendwie mitzuführen. Da PSW eine erweiterte SCENE ist (unter anderem), wird einfach ein PauseMenu aufgerufen, zuerst aber das PSW als last_scene von UC_INTERFACE gespeichert. Wenn man dann im Menu auf resume game geht, wird:

- der event_loop vom Menu gestoppt
- als nächste Szene wieder das PSW gesetzt
- der event_loop von PSW wieder zum laufen gebracht
- eine boolean Variable is_initialized auf True gesetzt

Der 4. Punkt ist deshalb notwendig, weil eine ESDL_APPLICATION ja eine Szene nach der anderen ausführt. Wenn ich also als nächste Szene ein PSW setze, wird automatisch das deferred feature initialize_scene der betreffenden ESDL_SCENE ausgeführt. Da initialize_scene von PSW jedoch alles macht (von Bild schneiden über workspaces bilden bis Sound neu abspielen). Deshalb wurde die varuable is_initialized eingeführt, um so die benötigten features neu zu initialisieren (aus irgendeinem Grund müssen die Buttons neu initialisiert werden und natrlich alle Events) und den Zustand der restlichen so zu behalten.

3.6. Klassen

3.6.1. cutter/QUAD_CUTTER.e

```
indexing
    description: "Factory to cut images into a puzzle"
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    QUAD_CUTTER

inherit

    ESDL_SHARED_BITMAP_FACTORY

create

    make
            -- initialize cutter


feature -- Initialization

    make
            -- initialize cutter


feature -- Factory

    make_from_image (image_name: STRING; number_of_parts: INTEGER)
            -- create number_of_parts of image given by image_name
            (name of file)
        require
            name_valid: image_name /= Void and then not image_name.
            is_empty
        ensure
            puzzle_created: last_puzzle /= Void


feature -- Access

    last_puzzle: UC_PUZZLE
        -- the puzzle that was cutted

    last_image: ESDL_BITMAP
        -- and the image, that was used to cut it


feature {NONE} -- Implementation

    calculate_num_p(number_of_parts: INTEGER) : TUPLE[INTEGER,
    INTEGER] is
            -- calculate number of parts in row and col
            require
                image_not_void: image /= void
```

```
            meaningfull_number: number_of_parts > 0
        nsure
            result_created: result /= Void
            result_meaningful:  result.integer_item (1) > 0 and
             result.integer_item(2) > 0
            result_in_intervall_lower: (result.integer_item (1)
             * result.integer_item(2)) > 0.9*number_of_parts
            result_in_intervall_upper: (result.integer_item (1)
             * result.integer_item(2)) < 1.1*number_of_parts
            ratio_reflects_image: true -- todo

    calculate_part_size(row_col_number: TUPLE[INTEGER, INTEGER]) :
    TUPLE[INTEGER, INTEGER] is
        -- calculate size of parts
        require
            image_not_void: last_image /= void
            input_meaningful: row_col_number /= Void and then (
             row_col_number.integer_item(1) > 0 and
            row_col_number.integer_item(2) > 0 )
            ratio_reflects_image: true -- todo
        ensure
            result_created: result /= Void
            result_meaningful: result /= Void and then ( result
            .integer_item(1) > 0 and result.integer_item(2) > 0
             )
            size_somewhat_quadratic_1: result.integer_item(1) <
             2* result.integer_item(2)
            size_somewhat_quadratic_2: result.integer_item(2) <
             2* result.integer_item(1)

    create_parts(row_col_number: TUPLE[INTEGER, INTEGER]; size:
    TUPLE[INTEGER, INTEGER]) is
        -- does the actual cutting
        require
            image_not_void: last_image /= void
            last_puzzle_not_void: last_puzzle /= void
            number_useful: row_col_number /= Void and then (
            row_col_number.integer_item(1) > 0 and
            row_col_number.integer_item(2) > 0 )
            size_useful: size /= Void and then ( size.
            integer_item(1) > 0 and size.integer_item(2) > 0 )
        ensure
            puzzle_created: last_puzzle /= void

    scale(x,y : INTEGER) : DOUBLE is
        -- calculate zoom-factor such that x*y fits in
        max_width*max_height


feature {NONE} -- constants

    max_width : INTEGER is 400
    max_height : INTEGER is 600
        -- this denotes the maximal size, the image should have
        -- the image is zoomed to fit in this area


end -- class QUAD_CUTTER
```

3.6.2. cutter/SHARED_QUAD_CUTTER.e

```
indexing
    description: "Shared access to cutter singleton."
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    SHARED_QUAD_CUTTER


feature -- Singleton Access

    cutter: QUAD_CUTTER is
        -- cutter singleton
        once
            create Result.make
        ensure
            cutter_not_void: Result /= Void


end -- class SHARED_QUAD_CUTTER
```

3.6.3. gui_elements/BUTTON.e

```
indexing
    description: "Objects that implements a button"
    author: "Beat Steiger and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    BUTTON

inherit
    ESDL_DRAWABLE_CONTAINER[ESDL_DRAWABLE]
        rename
            make as make_container,
            initialize_events as initialize_container_events
        end

    UC_SHARED_ITEMS
        rename
            width as window_width,
            height as window_height
        export
            {NONE} all
        undefine
            copy,
            is_equal
        end
```

```
creation
    make,
    make_with_name,
    make_with_icon

feature -- Access

    font : ESDL_FONT

    active_color, normal_color, prelight_color, highlight_color :
    ESDL_COLOR

    border_color : ESDL_COLOR

    border_width : INTEGER

    name : STRING

    icon : ESDL_BITMAP

    min_width, min_height : INTEGER

    horizontal_margin, vertical_margin : INTEGER

    button_clicked_event : EVENT_TYPE [TUPLE []]

    mouse_over_button_event : EVENT_TYPE [TUPLE []]

    has_icon, has_name : BOOLEAN

    corners_rounded : BOOLEAN

    is_active, change_color_on_toggle_active : BOOLEAN

    is_prelighted, is_highlighted, is_pressed : BOOLEAN

feature -- Status setting

    set_width (a_width : INTEGER) is
            -- set the 'width' of the button
        require
            not_too_small : a_width >= min_width

    set_height (a_height : INTEGER) is
            -- set the 'height' of the button
        require
            not_too_small : a_height >= min_height

    set_vertical_margin (a_vertical_margin : INTEGER) is
            -- set the 'vertical_margin' of the button
        require
            not_negative : a_vertical_margin >= 0

    set_horizontal_margin (a_horizontal_margin : INTEGER) is
            -- set the 'horizontal_margin' of the button
        require
            not_negative : a_horizontal_margin >= 0
```

```
    set_name (a_name : STRING) is
            -- set the 'name' of the button
        require
            a_name_not_void: a_name /= void

    set_icon (an_icon : ESDL_BITMAP) is
            -- set the 'icon' of the button
        require
            an_icon_not_void: an_icon /= void

    set_font (a_font : ESDL_FONT) is
            -- set the 'font' of the button
        require
            a_font_not_void: a_font /= void

    set_normal_color (a_normal_color : ESDL_COLOR) is
            -- set the 'normal_color' of the button
        require
            a_normal_color_not_void: a_normal_color /= void

    set_prelight_color (a_prelight_color : ESDL_COLOR) is
            -- set the 'prelight_color' of the button
        require
            a_prelight_color_not_void: a_prelight_color /= void

    set_highlight_color (a_highlight_color : ESDL_COLOR) is
            -- set the 'highlight_color' of the button
        require
            a_highlight_color_not_void: a_highlight_color /= void

    set_border_color (a_border_color : ESDL_COLOR) is
            -- set the 'border_color' of the button-border
        require
            a_border_color_not_void: a_border_color /= void

    set_border_width (a_border_width : INTEGER) is
            -- set the 'border_width' to 'a_border_width'
        require
            not_too_small: a_border_width >= 0

    set_corners_rounded is
            -- make the corners round

    set_corners_square is
            -- make the corners square

    toggle_active is
            -- toggle 'active' state of this button.
            -- right after creation the state is false

    toggle_change_color_on_toggle_active is
            -- toggle 'change_color_on_toggle_active' state of this
              button.
            -- right after the creation the state is true


feature {NONE} -- Creation
```

```
    make (an_event_loop : ESDL_EVENT_LOOP) is
            -- create a button
        require
            an_event_loop_not_void : an_event_loop /= void

    make_with_name (a_name : STRING; an_event_loop :
    ESDL_EVENT_LOOP) is
            -- create a button with the 'name' set to 'a_name'
        require
            a_name_not_void : a_name /= void
            an_event_loop_not_void : an_event_loop /= void

    make_with_icon (an_icon : ESDL_BITMAP; an_event_loop :
    ESDL_EVENT_LOOP) is
            -- create a button with 'an_icon' on it
        require
            an_icon_not_void : an_icon /= void
            an_event_loop_not_void : an_event_loop /= void


feature {NONE} -- Implementation

    drawable_name : ESDL_STRING

    icon_cell : DS_BILINKABLE [ESDL_DRAWABLE]

    body : ESDL_RECTANGLE

    event_loop : ESDL_EVENT_LOOP

    set_default_values is
            -- set some values to their default

    initialize_events is
            -- initialize requred mouse_events

    button_clicked (ev : ESDL_MOUSEBUTTON_EVENT) is
            -- perform button-clicked actions

    button_released (ev : ESDL_MOUSEBUTTON_EVENT) is
            -- perform button-released actions

    button_prelight (ev : ESDL_MOUSEMOTION_EVENT) is
            -- perform prelight-button actions when mouse is over
            button


end -- class BUTTON
```

3.6.4. gui_elements/TEXT_FIELD.e

```
indexing
    description: "Object that implements a TextField"
    author: "Steiger Beat"
```

```
    date: "Fri 10 June 2005"

class
    TEXT_FIELD

inherit
    UC_SHARED_ITEMS
        rename
            width as screen_width,
            height as screen_height
        export
            {NONE} all
        undefine
            copy,
            is_equal,
            default_create
        end

    ESDL_DRAWABLE_CONTAINER [ESDL_DRAWABLE]
        rename
            make as make_container
        end

    ESDL_ANIMATABLE
        undefine
            default_create,
            copy,
            is_equal
        end


creation
    make


feature -- Access

    background_color : ESDL_COLOR
        -- the color of the background

    cursor_color : ESDL_COLOR
        -- the color of the cursor

    border_color : ESDL_COLOR
        -- the color of the border

    value : STRING
        -- the string-value of the text-field

    border_width : INTEGER
        -- the width of the border

    font : ESDL_FONT
        -- the font in which the text will be displayed

    focused : BOOLEAN
        -- is the text-field focused?
```

```
    cursor_blinkrate : INTEGER
        -- the blinking rate of the cursor
        -- (eg a cursor_blinkrate of 2 means that the cursor blinks
          twice a second)

    return_key_pressed_event : EVENT_TYPE [TUPLE[STRING]]

    set_font (a_font : ESDL_FONT) is
            -- sets the 'font' of the TEXT_FIELD object to 'a_color
            '
        require

    set_background_color (a_color : ESDL_COLOR) is
            -- sets the 'background_color' of the TEXT_FIELD object
             to 'a_color'
        require
            a_color_exists : a_color /= void
        do

    set_cursor_color (a_color : ESDL_COLOR) is
            -- sets the 'cursor_color' of the cursor to 'a_color'
        require
            a_color_exists : a_color /= void

    set_border_color (a_color : ESDL_COLOR) is
            -- sets the 'border_color' of the TEXT_FIELD object to
            'a_color'
        require
            a_color_exists : a_color /= void

    set_border_width (a_border_width : INTEGER) is
            -- sets the 'border_width' of the TEXT_FIELD object to
            'a_border_width'

    set_width (a_width : INTEGER) is
            -- sets the 'width' of the TEXT_FIELD object to '
            a_width'
        require
            width_not_too_small: width >= border_width

    set_cursor_blinkrate (br : INTEGER) is
            -- sets the 'cursor_blinkrate' of the cursor to '
            a_cursor_blinkrate'
        require
            blinkrate_not_zero : br /= 0

    set_value (v : STRING) is
            -- sets the 'value' of the TEXT_FIELD object to '
            a_value'
        require
            v_not_void : v /= void

    go_to_time (a_time: INTEGER) is
            -- Go to the frame corresponding to 'a_time' (in
            milliseconds) of the 'cursor' animation.


feature {NONE} -- Initialisation
```

```
    make (an_event_loop : ESDL_EVENT_LOOP) is
            -- Create a textfield with default values
        require
            an_event_loop_not_void : an_event_loop /= void


feature {NONE} -- Implementation

    field : ESDL_RECTANGLE

    cursor : ESDL_SPRITE

    anim : ARRAY[ESDL_POLYLINE]

    e_string : ESDL_STRING

    event_loop : ESDL_EVENT_LOOP

    keyboard : ESDL_KEYBOARD

    subscribed_procedures : LINKED_LIST[PROCEDURE[ANY,TUPLE]] is
            -- linked-list to hold agents which were subscribed to
            the
            -- key_down_event of the 'event_loop'
        once
            Result := create {LINKED_LIST[PROCEDURE[ANY,TUPLE]]}.
            make
        end

    set_focused (ev : ESDL_MOUSEBUTTON_EVENT) is
            -- set the focus on this textfield if left-mouse-button
            -- is clicked on it

    set_unfocused (ev : ESDL_MOUSEBUTTON_EVENT) is
            -- set the focus away from this textfield if left-mouse
            -button
            -- is clickt somewhere but on this textfield.

    get_key (ev : ESDL_KEYBOARD_EVENT) is
            -- Capture pressed keys

    display_value is
            -- generate the 'e_string' which will be displayed in
            the textfield

    is_modifier_key (a_key_code : INTEGER) : BOOLEAN is
            -- returns true if 'a_key_code' denotes a modifier_key
            (alt, ctrl, etc.)
        do


end -- class TEXT_FIELD
```

3.6.5. menus/CREDITS.e

```
indexing
    description: "Objects that represents the credits menu"
    author: "Benno Baumgartner, benno@student.ethz.ch"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    CREDITS

inherit

    MENU
        redefine
            initialize_scene
        end

    UC_SHARED_ITEMS
        export
            {NONE} all
        undefine
            default_create
        end


feature

    initialize_scene
            -- Initialize the scene


feature {NONE} -- Event handler

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here


end -- class CREDITS
```

3.6.6. menus/LOAD_GAME.e

```
indexing
    description: "load a game"
    author: "Markus Neidhart and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    LOAD_GAME

inherit
    MENU
        redefine
            initialize_scene
        end
```

```
feature -- Initialize

    load: UC_LOAD_PUZZLE

    initialize_scene is
            -- Initialize the scene

    set_situation (sit: PUZZLE_START_WINDOW) is
            -- set situation to 'sit'
        require
            sit_not_void: sit /= Void
        ensure
            situation_set: situation = sit



feature {NONE} -- Implementation

    situation: PUZZLE_START_WINDOW
            -- actual situation, will be set in UC_INTERFACE, which
              have it as last_scene

    text_field: TEXT_FIELD

    load_button: BUTTON

    psw: PUZZLE_START_WINDOW


feature {NONE} -- Event handler

    load_puzzle (val: STRING) is
            -- load the puzzle
        require
            val_not_Void: val /= Void

    load_button_clicked is
            -- click on load button

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here


feature -- Input checking

    input_check : INTEGER is
            -- checks given input and returns error_code

    error_message(error_code: INTEGER) is
            -- put error message to given error_code on screen

    error_string : ESDL_STRING
            -- this is the error message


end -- class LOAD_GAME
```

3.6.7. menus/MENU.e

```
indexing

    description: "[
                  MENU contains functionalities used by all menus
                   in the
                  game.
                ]"
    author: "Benno Baumgartner, benno@student.ethz.ch, Mouse-
    Support by Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

deferred class

    MENU

inherit

    ESDL_SCENE
        redefine
            handle_key_down_event, handle_mouse_motion_event,
            handle_mouse_button_up_event
        end

    UC_SHARED_ITEMS
        undefine
            default_create
        end


feature

    initialize_scene is
            -- Initialize the scene


feature {NONE} -- Event handler

    handle_key_down_event (a_keyboard_event: ESDL_KEYBOARD_EVENT)
    is
            -- Handle keyboard events.

    handle_mouse_motion_event (mme: ESDL_MOUSEMOTION_EVENT) is
            -- handle mouse motion

    handle_mouse_button_up_event (a_mouse_button_event:
    ESDL_MOUSEBUTTON_EVENT) is
            -- handle mouse click

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here
        deferred
```

```
        end

feature {MENU,PUZZLE_START_WINDOW} -- Implementation

    scene: ESDL_DRAWABLE_CONTAINER [ESDL_DRAWABLE]
            -- The scene

    last_scene: ESDL_SCENE

    set_last_scene(ls: ESDL_SCENE) is
            -- create an instruction menu

    set_next_scene(ns: ESDL_SCENE) is
            -- set the next scene

    select_font: ESDL_BITMAPFONT
            -- The font used to display a selected menu entry

    normal_font: ESDL_BITMAPFONT
            -- The font used to display the not selected menu
             entries

    menu_entries: ARRAY [ESDL_STRING]
            -- The menu entries

    selected_nr: INTEGER
            -- The number of the selected entry

    update_menu_entries is
            -- Updates the 'menu_entry'

    add_menu_entries_to_scene is
            -- Add menu entries to 'scene'.

    restart_event_loop is
            -- restart the event loop

    empty_main_container is
            -- empty the main container


end -- Class Menu
```

3.6.8. menus/NEW_GAME.e

```
indexing
    description: "lets the user set an image and a number of parts
    and starts the game with these parameters"
    author: "Markus Neidhart and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    NEW_GAME
```

```
inherit
    MENU
        redefine
            initialize_scene
        end

    ESDL_SHARED_BITMAP_FACTORY
        undefine
            copy, is_equal, default_create
        end


feature -- Initialize

    save: UC_SAVE_PUZZLE

    initialize_scene is
            -- Initialize the scene


feature {NONE} -- Implementation

    situation: PUZZLE_START_WINDOW
            -- actual situation, will be set in UC_INTERFACE, which
              have it as last_scene

    text_field: TEXT_FIELD

    text_field2:TEXT_FIELD

    start_button: BUTTON

    psw: PUZZLE_START_WINDOW


feature {NONE} -- Event handler

    start_button_clicked is
            -- click on start button

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here


feature -- Input checking

    input_check : INTEGER is
            -- checks given input and returns error_code

    error_message(error_code: INTEGER) is
            -- put error message to given error_code on screen

    error_string : ESDL_STRING
            -- this is the error message


end -- class NEW_GAME
```

3.6.9. menus/PREDEFINED_GAME.e

```
indexing
    description: "play a predefined game"
    author: "Markus Neidhart and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    PREDEFINED_GAME

inherit
    MENU
        redefine
            initialize_scene
        end


feature -- Initialize

    initialize_scene is
            -- Initialize the scene


feature {NONE} -- Event handler

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here


end -- class OPTIONS
```

3.6.10. menus/SAVE_GAME.e

```
indexing
    description: "Objects that saves the uc puzzle"
    author: "Markus Neidhart and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    SAVE_GAME

inherit
    MENU
        redefine
            initialize_scene
        end


feature -- Initialize
```

```
    save: UC_SAVE_PUZZLE

    initialize_scene is
            -- Initialize the scene

    set_situation (sit: PUZZLE_START_WINDOW) is
            -- set situation to 'sit'
        require
            sit_not_void: sit /= Void
        ensure
            situation_set: situation = sit


feature {NONE} -- Implementation

    situation: PUZZLE_START_WINDOW
        -- actual situation, will be set in UC_INTERFACE, which
        have it as last_scene

    text_field: TEXT_FIELD

    save_button: BUTTON


feature {NONE} -- Event handler

    save_puzzle (val: STRING) is
            -- save the puzzle
        require
            val_not_Void: val /= Void

    save_button_clicked is
            -- click on save button

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here


end -- class SAVE_GAME
```

3.6.11. menus/SELECT_GAME.e

```
indexing
    description: "user choose if play a game with his own
    parameters or a fixed modus game"
    author: "Markus Neidhart and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    SELECT_GAME

inherit
    MENU
        redefine
```

```
            initialize_scene
        end


feature -- Initialize

    initialize_scene is
            -- Initialize the scene


feature {NONE} -- Event handler

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here


end -- class SELECT_GAME
```

3.6.12. menus/UC_INTERFACE.e

```
indexing
    description: "Main Menu for Game UC."
    author: "Team UC with thanks to Benno Baumgartner"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_INTERFACE

inherit

    MENU
        redefine
            initialize_scene
        end


feature -- commands

    initialize_scene is
            -- Initialize the scene


feature {NONE} -- Event handler

    on_select is
            -- Set the 'next_scene' according to 'selected_nr' here


end
```

3.6.13. model_view/UC_PUZZLE.e

```
indexing
    description: "This contains all puzzle_parts of a puzzle."
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_PUZZLE

create
    make_empty


feature -- initialization
    make_empty is
            -- creates an empty puzzle.
            -- use a cutter to make puzzles


feature -- Access

    get_puzzle_part (pos_x: INTEGER; pos_y: INTEGER):
    UC_PUZZLE_PART is
            -- get the puzzle_part with pos_x and pos_y, Void when
            not found

    image_width: INTEGER
            -- width of the image, this puzzle was made of

    image_height: INTEGER
            -- height of the image, this puzzle was made of

    part_width: INTEGER
            -- width of one part of this puzzle

    part_height: INTEGER
            -- height of one part of this puzzle

    num_parts_width: INTEGER
            -- number of parts in width

    num_parts_height: INTEGER
            -- number of parts in height


feature {QUAD_CUTTER} -- Status setting

    set_image_size (size : TUPLE[INTEGER, INTEGER]) is
            -- set size of the image to size=(width, height)
            require
                size_not_void: size /= void

    set_part_size (size : TUPLE[INTEGER, INTEGER]) is
            -- set size of the parts to size=(width, height)
            require
                size_not_void: size /= void

    set_part_num (num : TUPLE[INTEGER, INTEGER]) is
```

```
            -- set number of parts to num=(num in width, num in
            height)
            require
                num_not_void: num /= void


feature {QUAD_CUTTER} -- Element change

    extend (new_part : UC_PUZZLE_PART) is
            -- put new part in part_list
            require
                new_part_not_void: new_part /= void


feature -- Element Change
    mix_up is
            -- mixes given list up


feature {UC_PUZZLE_VIEW} -- Parts

    part_list: DS_BILINKED_LIST[ESDL_DRAWABLE]
            -- the list of all puzzle_parts in this puzzle


end -- class UC_PUZZLE
```

3.6.14. model_view/UC_PUZZLE_PART.e

```
indexing
    description: "This class denotes the model of the puzzle parts
    in game UC."
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_PUZZLE_PART

inherit
    ESDL_DRAWABLE
        redefine default_create, is_equal end
        -- altough a puzzle_part is itself drawable, one should use
         a uc_puzzle_part_view to actually draw a uc_puzzle_part

    ESDL_SHARED_BITMAP_FACTORY
        export {NONE} all
        undefine default_create, copy, is_equal end
        -- we need the bitmapfactory to rotate the pictures

create
    make_from_bitmap


feature {NONE} -- Creation
```

```
    default_create is
            -- initialize object
            -- DO NOT USE THIS ONE

    make_from_bitmap(bitmap: ESDL_BITMAP) is
            -- initialize object


feature -- Comparison

    is_equal (other : like current) : BOOLEAN is
            -- tow puzzle_parts are equal, if they show the same
            part of the picture


feature -- Status

    width: INTEGER
            -- width of this part

    height: INTEGER
            -- height of this part

    pos_x : INTEGER
            -- position of this part in the whole puzzle
            -- coords start at top right
            -- pos_x gives col

    pos_y : INTEGER
            -- position of this part in the whole puzzle
            -- coords start at top left
            -- pos_y gives row

    locked: BOOLEAN
            -- may this part be moved, that means, is part already
            placed correctly ?

    correct(ax,ay:INTEGER): BOOLEAN is
            -- is this part at its correct place ?
            -- x,y denote the top left corner of to_field


feature -- Drawing

    draw(a_surface: ESDL_SURFACE) is
            -- draw part on a_surface
            -- for compatibility. use a view of this part instead
            and then call draw_oriented

    draw_oriented(a_surface: ESDL_SURFACE; orient,an_x,an_y :
    INTEGER) is
            -- draw this part on a_surface with given orient
            require
                orient_feasible: (orient>0 and orient<5)
                surface_there: a_surface /= void
```

```
feature -- Element change

    lock  is
            -- lock this part

    set_pos (an_x, an_y: INTEGER) is
            -- set pos of this part in the puzzle


feature -- Pictures
    picture: ESDL_BITMAP
        -- the picture of the part

    -- following singleton access to rotated images
    picture2: ESDL_BITMAP is

    pic2: ESDL_BITMAP

    picture3: ESDL_BITMAP is

    pic3: ESDL_BITMAP

    picture4: ESDL_BITMAP is

    pic4: ESDL_BITMAP


invariant
    pic_not_void : picture /= void


end -- class UC_PUZZLE_PART
```

3.6.15. model_view/UC_PUZZLE_PART_VIEW.e

```
indexing
    description: "This is a view of a puzzle_part.."
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_PUZZLE_PART_VIEW

inherit
    ESDL_DRAWABLE
        redefine is_equal end

create
    make_from_puzzle_part


feature -- Initialization

    make_from_puzzle_part(puzz_part: UC_PUZZLE_PART) is
```

```
            -- create a new view of given puzz_part
        require
            puzz_part_not_void: puzz_part /= void


feature -- Comparison

    is_equal (other: like current) : BOOLEAN is
            -- two puzzle_part_views are equal if they show the
            same part


feature -- Status report

    save: STRING is
            -- give all the information that are required to save
            this ppv

    orientation: INTEGER
        -- how is this view oriented, 1 is normal, 2 rotated by 90
        deg clockwise usw

    width: INTEGER is
            -- width of this view

    height: INTEGER is
            -- height of this part

    locked: BOOLEAN is
            -- is part locked

    correct(ax, ay: INTEGER): BOOLEAN is
            -- is part correctly set
            -- ax, ay denote coordination transformation


feature -- Element change

    rotate is
            -- rotate view

    lock is
            -- lock this part


feature -- Drawing

    draw(a_surface: ESDL_SURFACE) is
        -- draw part on a_surface

    draw_solved(a_surface: ESDL_SURFACE; ax,ay : INTEGER) is
            -- draw part on a_surface
            require
                puzzle_part.locked


feature -- Implementation
```

```
    puzzle_part: UC_PUZZLE_PART
        -- the part this view shows


end -- class UC_PUZZLE_PART_VIEW
```

3.6.16. model_view/UC_PUZZLE_VIEW.e

```
indexing
    description: "this is a view of a puzzle."
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_PUZZLE_VIEW

inherit
    ESDL_DRAWABLE_CONTAINER[ESDL_DRAWABLE]

create
    make_from_puzzle, make_from_other, make


feature -- Initialization
    make_from_other(other: UC_PUZZLE_VIEW) is
            -- initialize this puzzle_view
            -- copy the parts-list from given puzzle_view
            require
                other_not_void: other /= void

    make_from_puzzle(a_puzzle: UC_PUZZLE) is
            -- initialize this puzzle_view with given puzzle
            require
                a_puzzle_is_not_void: a_puzzle /= void


feature -- Status

    parts_in_row : INTEGER
        -- how many parts are drawn in a row ?

    copy_parameter_from(other: UC_PUZZLE_VIEW) is
            -- set the parameters according to other


feature -- Element change

    set_parts_in_row (i: INTEGER) is
            -- set parts_in_row to i
            require
                i_useful : i > 0


feature -- puzzle attributes
```

```
    image_width : INTEGER
    image_height: INTEGER
        -- size of the image this view was made of

    part_width: INTEGER
    part_height: INTEGER
        -- size of the parts this puzzle was made of


end -- class UC_PUZZLE_VIEW
```

3.6.17. sound/ESDL_AUDIO_CONSTANTS.e

```
indexing
    description: "Some audio constants"
    author: "Yann Mueller muelleya@student.ethz.ch"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    ESDL_AUDIO_CONSTANTS

feature
    Esdl_audio_format_u8: INTEGER is
            -- Unsigned 8-bit samples
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_U8"
        end

    Esdl_audio_format_s8: INTEGER is
            -- Signed 8-bit samples
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_S8"
        end

    Esdl_audio_format_u16lsb: INTEGER is
            -- Unsigned 16-bit samples
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_U16LSB"
        end

    Esdl_audio_format_s16lsb: INTEGER is
            -- Signed 16-bit samples
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_S16LSB"
        end
```

```
    Esdl_audio_format_u16msb: INTEGER is
            -- As above, but big-endian byte order
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_U16MSB"
        end

    Esdl_audio_format_s16msb: INTEGER is
            -- As above, but big-endian byte order
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_S16MSB"
        end

    Esdl_audio_format_u16: INTEGER is
            --
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_U16"
        end

    Esdl_audio_format_s16: INTEGER is
            --
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_S16"
        end

    Esdl_audio_format_u16sys: INTEGER is
            --
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_U16SYS"
        end

    Esdl_audio_format_s16sys: INTEGER is
            --
        external
            "C macro use <esdl.h>"
        alias
            "AUDIO_S16SYS"
        end


end
```

3.6.18. sound/ESDL_MIXER.e

```
indexing
```

```
    description: "audio mixer device - OO wrapper for SDL_mixer"
    author: "Yann Mueller muelleya@student.ethz.ch"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    ESDL_MIXER

inherit

    ESDL_CONSTANTS
        export
            {NONE} all
        end

    ESDL_AUDIO_CONSTANTS
        export
            {NONE} all
        end

    SDL_MIXER_FUNCTIONS
        export
            {NONE} all
        end

    SDL_FUNCTIONS_EXTERNAL
        export
            {NONE} all
        end

create

    make


feature {NONE} -- Initialization

    make is
            -- Create a new mixer instance.


feature -- Basic operations

    open is
            -- Open the mixer device
        require
            not_open: not is_open

    close is
            -- Close the mixer device
        require
            is_open

    play_music is
            -- Play the music
        require
            device_open: is_open
            music_set: music /= Void
```

```
        stop_music is
                -- Stop the music
            require
                device_open: is_open
                music_set: music /= Void

        pause_music is
                -- pause the music
            require
                device_open: is_open
                music_set: music /= Void

        resume_music is
                -- resume paused music
            require
                device_open: is_open
                music_set: music /= Void

        halt_channel(a_channel :INTEGER) is
                -- halt channel


    feature -- Access

        music_paused: BOOLEAN
                -- is the music currently paused?

        music: ESDL_MUSIC
                -- the current music object

        music_volume: INTEGER
                -- the volume of the music

        frequency: INTEGER
                -- The mixer frequency

        bits_per_sample: INTEGER
                -- bits per sample

        channels: INTEGER
                -- number of channels

        chunksize: INTEGER
                -- chunk size of the mixer


    feature -- Element change

        set_music (a_music: ESDL_MUSIC) is
                -- Set the music object

        set_music_volume (a_volume: INTEGER) is
                -- Set the music volume
            local
                i: INTEGER

        set_frequency (a_frequency: INTEGER) is
```

```
            -- set mixer frequency
        require
            not_open: not is_open

    set_bits_per_sample (a_bits_per_sample: INTEGER) is
            -- set bits per sample
        require
            not_open: not is_open
            valid: a_bits_per_sample = 8 or a_bits_per_sample = 16

    set_channels (a_channels: INTEGER) is
            -- set number of channels
        require
            not_open: not is_open
            valid: a_channels = 1 or a_channels = 2

    set_chunksize (a_chunksize: INTEGER) is
            -- set the chunk size
        require
            not_open: not is_open


feature -- Status report

    is_open: BOOLEAN
            -- Is the device open

    is_music_playing: BOOLEAN is
            -- Is the music playing


end
```

3.6.19. sound/ESDL_MUSIC.e

```
indexing
    description: "Music class to play in the mixer"
    author: "Yann Mueller muelleya@student.ethz.ch"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    ESDL_MUSIC

inherit

    SDL_MIXER_FUNCTIONS

create

    make_from_file (a_filename: STRING)
            -- Create a music object from a file


feature {NONE} -- Initialization
```

```
    make_from_file (a_filename: STRING) is
            -- Create a music object from a file


feature -- Access

    loops: INTEGER
            -- loop count. 0 means play only once. -1 means loop
            forever

    Sizeof: INTEGER is 0
            -- unknown size -> 0


feature -- Element change

    rewind
            -- rewind_music

    set_loops (a_loops: INTEGER)
            -- set loop count
        ensure
            set: loops = a_loops


feature -- Removal

    dispose
            -- Free external resources


feature {ESDL_MIXER}

    music_handle: POINTER


feature -- Implementation

    filename: STRING


end -- class ESDL_MUSIC
```

3.6.20. sound/MUSIC_PLAYER.e

```
indexing
    description: "Objects that manages all the sound events"
    author: "Florian Keusch, fkeusch@student.ethz.ch, Markus
    Neidhart, nemarkus@student.ethz.ch and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    MUSIC_PLAYER
```

```
create
    make


feature -- Initialization

    make is
            -- create 'musictrack'
        ensure
            musictrack_exists: musictrack /= Void
            musictrack_is_open: musictrack.is_open


feature -- player functions

    open_sound (track: STRING) is
            -- opens the soundfile specified by 'track'. Has to be
            a .ogg file.

    play () is
            -- plays loaded soundfile
        require
            sound_file_loaded: musictrack.music /= Void
        ensure
            music_is_playing: musictrack.is_music_playing

    stop () is
            -- stop loaded soundfile
        require
            sound_file_loaded: musictrack.music /= Void
        ensure
            sound_file_is_void: musictrack.music = Void


    pause () is
            -- pause loaded soundfile if playing
            -- unpause loaded soundfile if stopped
        require
            sound_file_loaded: musictrack.music /= Void

    next () is
            -- plays the next song

    previous () is
            -- plays the previous song

    set_volume (vol: INTEGER) is
            -- sets the volume
        require
            vol_not_void: vol /= Void

    turn_up is
            -- set the volume louder

    turn_down is
            -- turns the volume down
```

```
feature -- Implemenation

    read_sound_dir is
            -- go through sound_dir and check for playable files ->
              put into sound_container

    play_song (a_string: STRING) is
            -- plays the song with the string
            -- caution: string must exist

    set_sound_dir (a_string: STRING) is
            -- set a new sound_dir and read it
        require
            not_void_string: a_string /= Void

    next_song: STRING is
            -- gives back the string of the next song

    previous_song: STRING is
            -- gives back the string of the previous song

    set_event_loop(an_ev: ESDL_EVENT_LOOP) is
            -- set new event loop

    check_terminated:BOOLEAN is
            -- checks if song is finished

    check_next is
            -- checks if song is finished -> and then plays next

    mute is
            -- set volume to 1
            -- backup old volume


feature -- Status

    is_playing: BOOLEAN is
            -- checks if player is playing

    is_stopped: BOOLEAN is
            -- checks if player is stopped

    is_paused: BOOLEAN is
            -- checks whether player is paused

    song_nr: INTEGER is
            -- get the song number

    song_title: STRING is
            -- get the song title
            -- with path

    volume: INTEGER is
            -- gives back the music volume

    song_title_cut: STRING is
```

```
            -- get the song title cut

    is_empty: BOOLEAN is
            -- checks if player has items to play


feature -- Access

    musictrack: ESDL_MIXER
        -- Mixer object

    sound_dir: STRING
            -- string that indicates the sound directory

    sound_container: LINKED_LIST[STRING] is
                -- sound_container of strings
            once
                create Result.make
            ensure
                not_void: Result /= void
            end

    playing: INTEGER
            -- number of the song that is currently played or
            stopped

    ready: BOOLEAN
            -- indicates whether player is ready (sound_container
            has at least one element)

    event_loop_set: BOOLEAN
            -- check if event_loop is set

    ev: ESDL_EVENT_LOOP
            -- the event_loop that was given

    old_volume: INTEGER


invariant

    musictrack_exists: musictrack /= Void
    playing >= 0
    volume >= 0
    old_volume >= 0


end -- class MUSIC_PLAYER
```

3.6.21. workspace/UC_FROM_SPACE

```
indexing
    description: "This denotes a special kind of puzzle_view. Parts
     in this view only disappear, when they are locked.."
    author: "Stefan Hildenbrand and Team UC"
```

```
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_FROM_SPACE

inherit

    UC_WORKSPACE
        redefine
            cond_delete
        end

    ESDL_SHARED_BITMAP_FACTORY
        export {NONE} all
        undefine copy, is_equal
        end

create
    make_from_puzzle


feature -- Element change
    cond_delete(v: ESDL_DRAWABLE) is
            -- redefine delete to handle the case where we want to
            delete
            -- the hidden element (which will most likely happen
            very often)
            -- IMPORTANT: this does not delete the part_view if it
            is not locked yet!
            --     instead it makes a copy of the part to delete and
             keeps the copy !


end -- class UC_FROM_SPACE
```

3.6.22. workspace/UC_TO_SPACE.e

```
indexing
    description: "This is a special view of a puzzle. This view
    does not allow deletion and draws the parts always correct."
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_TO_SPACE

inherit
    UC_PUZZLE_VIEW
        redefine draw end

create
    make
```

```
feature

    draw (a_surface: ESDL_SURFACE) is
            -- Draw 'Current' to 'a_surface'
            -- this draws the parts in this workspace in a sorted
            way


feature -- Status report

    save: STRING is
            -- give a string back that holds all information for
            saving this workspace


end -- class UC_TO_SPACE
```

3.6.23. workspace/UC_WORKSPACE.e

```
indexing
    description: "this is a special kind of puzzle_view, which
    supports sorted insertions, but does not impose restriction on
    extending or deleting"
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_WORKSPACE

inherit
    UC_PUZZLE_VIEW
        redefine
            draw, delete, width, height
        end

    ESDL_SHARED_BITMAP_FACTORY
        export {NONE} all
        undefine copy, is_equal
        end

create
    make_from_puzzle, make_from_other, make


feature -- Drawing

    draw (a_surface: ESDL_SURFACE) is
            -- Draw 'Current' to 'a_surface'
            -- this draws the parts in this workspace in a sorted
            way


feature -- Status Report
```

```
    save: STRING is
            -- give a string back that holds all information for
            saving this workspace
        ensure
            Result_not_Void: Result /= Void

    width: INTEGER is
            -- width of this workspace

    height: INTEGER is
            -- height of this workspace


feature -- Element change

    hidden_part: UC_PUZZLE_PART_VIEW
        -- remember the part we hide

    hidden_index: INTEGER
        -- where is the part we had hidden

    hide (part_view : UC_PUZZLE_PART_VIEW) is
            -- hide given part, but do not reorder parts yet
            require
                item_is_there: has(part_view)

    unhide is
            -- show the part that was hidden before
            -- this function si tolerant, if there is no part we
            hide, just do nothing
            --require
            --  hidden_part_exists: hidden_part /= void

    cond_delete(v: ESDL_DRAWABLE) is
            -- sometimes we do want to make an conditional delete
            require
                v_not_void: v/= void

    delete(v: ESDL_DRAWABLE) is
            -- redefine delete to handle the case where we want to
            delete
            -- the hidden element (which will most likely happen
            very often)

    extend_sorted(v: UC_PUZZLE_PART_VIEW) is
            -- put given part v in the list, such that it appears
            there, where the user moved it
            -- ax,ay are coordinates, the source is the top right
            corner of this container!
            require
                v_not_void: v /= void

    sortable (v : UC_PUZZLE_PART_VIEW) : BOOLEAN is
            -- is a part with given coordinates sortable in this
            workspace
            require
                v_not_void: v /= void
```

```
    set_list(new_list : DS_BILINKED_LIST[ESDL_DRAWABLE]) is
            -- copy given part_list to this workspace
            require
                argument_not_void : new_list /= void


end -- class UC_WORKSPACE
```

3.6.24. FINISH_SCREEN.e

```
indexing
    description: "Screen that shows the image that was solved and
    congratulates the user."
    author: "Stefan Hildenbrand and Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    FINISH_SCREEN

inherit
    ESDL_SIMPLE_SCENE

    UC_SHARED_ITEMS
        rename
            height as window_height,
            width as window_width
        undefine
            default_create, is_equal, copy
        end

    MEMORY
        undefine
            default_create, is_equal, copy
        end

create
    make_from_image


feature -- Access

    make_from_image(image: ESDL_BITMAP) is
            -- create this scene

    handle_continue is
            -- continue with menu


end -- class FINISH_SCREEN
```

3.6.25. PUZZLE_START_WINDOW.e

```
indexing

    description: "Main class of puzzle game UC"
    author: "Team UC"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class

    PUZZLE_START_WINDOW

inherit
    ESDL_SCENE
        redefine
            initialize_scene, handle_key_down_event
        end

    UC_SHARED_ITEMS
        export
            {NONE} all
        undefine
            default_create
        end

    MEMORY -- need some additional memory management
        undefine
            default_create
        end

    SDLKEY_ENUM_EXTERNAL -- need the key codes

        export
            {NONE} all
        undefine
            default_create
        end


feature {NONE} -- Initialization

    initialize_scene is
            -- Create the main application.

    initialize_events is
            -- initialize all events

    create_buttons is
            -- create all the buttons

    uninit_guis  is
                -- removes all the guis and checks many things


feature {UC_SAVE_PUZZLE, UC_LOAD_PUZZLE} -- Status

    image_file_name: STRING
            -- Name of image we want to display
```

```
    image: ESDL_BITMAP
            -- the image that was used to cut the puzzle

    number_of_parts: INTEGER
            -- The number of parts


feature {UC_SAVE_PUZZLE, UC_LOAD_PUZZLE} -- Workspaces, Puzzle usw

    puzzle : UC_FROM_SPACE
        -- this is the container for the puzzle parts at the left
        side of screen

    workspace1 : UC_WORKSPACE
    workspace2 : UC_WORKSPACE
        -- this are the container where the user can do what he
        wants to

    active_space : UC_WORKSPACE
        -- this is the container which is actually on top and
        active

    to_field: UC_TO_SPACE
        -- this is the container for the puzzle_parts which are
        placed correctly

    puzzle_part_view : UC_PUZZLE_PART_VIEW
        -- active item to move


feature {NONE} -- Buttons for workspace control

    button_fs : BUTTON
        -- this is the button representating the from_space

    button_w1 : BUTTON
    button_w2 : BUTTON
        -- this are the buttons representating the workspaces

    button_trash : BUTTON
        -- button for deleting a part from a workspace

    button_reset: BUTTON
        -- button for resetting a workspace

    button_image : BUTTON
        -- button for showing the solution

    button_cheat : BUTTON
        -- button for cheating

    y_coord: INTEGER is
        -- y_coord for the menu gui
        once
            Result := height - mute.height - song_title_to_show.
            height - 5
        end
```

```
    load: BOOLEAN
        -- is to load?


feature -- load feature

    set_load is
        -- sets to load


feature {NONE} -- Agents

    handle_mouse_button_down_on_item (an_item: ESDL_DRAWABLE;
    a_mouse_button_event: ESDL_MOUSEBUTTON_EVENT) is
            -- This activates the part the user clicked on

    handle_mouse_motion (a_motion_event: ESDL_MOUSEMOTION_EVENT) is
            -- this moves the part around on the screen
            -- if button is released:
            -- if it is at correct position, it gets locked
            -- if part is moved to other position in workspace, the
              workspace gets rearranged
            -- otherwise it jumps back to where it came from


feature {NONE} -- Agents for workspace control

        handle_ws_button (to : UC_WORKSPACE) is
                -- handle event when a workspace button is clicked
                require
                    to_not_void: to /= void
                    as_not_void: active_space /= void

        toggle_buttons is
                -- set buttons according to active_space

        handle_trash_button is
                -- handle deleting a part from active workspace

        handle_reset_button is
                -- handle resetting active workspace

        handle_image_button is
                -- handle showing solution

        handle_image_disappear(a_time : INTEGER) is
                -- hide image at given point in time

        handle_finish is
                -- handle the fact, that puzzle is solved

        handle_cheat is
                -- handle cheat request


feature {NONE} -- Agent for key handling
```

```
        handle_key_down_event (a_keyboard_event:
        ESDL_KEYBOARD_EVENT) is
            -- handle key down events
            -- ESC: go to menu
            -- q: quit application
            -- b: back to the previous song
            -- n: go to the next song
            -- m: mute / unmute
            -- c: cheat
            -- 0,1,2 switch workspaces
            -- d delete actual part if one clicked otherwise delete
              all parts from workspace
            -- r reset workspace
            -- i show image


feature {NONE} -- Agents for sound control

        toggle_stop is
                -- toggle 'player' stop

        toggle_play_pause is
                -- toggle 'player' play pause

        toggle_mute is
                -- toggle 'player' mute

        toggle_previous is
                -- toggle 'player' previous

        toggle_next is
                -- toggle 'player' next


feature {NONE} -- Sound control

    play_icon, pause_icon, stop_icon, next_icon, prev_icon,
    sound_icon, nosound_icon: ESDL_BITMAP
    play_pause, stop, next, prev, mute : BUTTON

    check_next_song is
            -- checks if next song is played

    enable_song_title_showing is
            -- enable song title showing

    disable_song_title_showing is
            -- disable to show the song title

    song_title: STRING

    song_title_to_show: ESDL_STRING
        -- song_title that is currently showed

    song_show: BOOLEAN
        -- intidates whether to show the song

    init_player_gui is
```

```
            -- create the gui (prev, play/pause, stop, next, mute)

feature -- access

    initialize_loaded is
                -- initialize this psw when things are already
                loaded

    set_from_space (fs: UC_FROM_SPACE) is
            -- set puzzle to fs
        require
            fs_not_Void: fs /= Void

    set_to_space (ts: UC_TO_SPACE) is
            -- set to_field to ts
        require
            ts_not_Void: ts /= Void

    set_workspaces (workspace_1: UC_WORKSPACE; workspace_2:
    UC_WORKSPACE) is
            -- set both workspaces
        require
            not_Void: workspace_1 /= Void and workspace_2 /= Void

    set_image_file_name (a_filename: STRING) is
            -- set image_file_name to a_filename
        require
            a_filename_not_Void: a_filename /= Void

    set_number_of_parts (nop: INTEGER) is
            -- set the number of parts


feature {MENU} -- menu features

    set_next_scene(ns: ESDL_SCENE) is
            -- set the next scene

    restart_event_loop is
            -- restart the event loop

    is_initialized: BOOLEAN -- is this puzzle_start_window already
    initialized?

    pause_force: BOOLEAN -- true if player.pause is manually set by
     the user

    set_initialized (b: BOOLEAN) is
            -- set is_initialized to b


end
```

3.6.26. RANDOM_NUMBER_GENERATOR.e

```
indexing
    description: "Objects that Generates random numbers"
    author: "Florian Keusch, fkeusch@student.ethz.ch"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    RANDOM_NUMBER_GENERATOR

create
    make


feature {NONE} -- Random Access

    rand: RANDOM
        -- random number gen

    timer: TIME
        -- time at initialization

    init_seed: INTEGER
        -- the init seed of the random number generator

    next_nr: INTEGER
        -- the number of the next random number


feature -- init

    make is
            -- initializes the random number generator

    remake (seed: INTEGER) is
            -- reinitialize with new seed


feature -- functions

    next_int_mod  (n: INTEGER) : INTEGER is
            -- get the next random integer value mod n
            -- if n is 101 you get numbers from 0 to 100
            -- NOTE: n should be prime to get good results
            -- for example n := 101
        require
            n_not_void: n /= Void

    next_int_mod_minus_1 (n:INTEGER) : INTEGER is
            -- the same as next_int_mode but without zero
        require
            n_not_void: n /= Void

    next_int: INTEGER is
            -- get the next random integer value


    next_double: DOUBLE is
            -- get the next random double value
```

```
            -- normalized between 0 and 1

    next_real: REAL is
            -- get the next random real value
            -- normalized between 0 and 1

    next_bool: BOOLEAN is
            -- get random boolean value


end -- class RANDOM_NUMBER_GENERATOR
```

3.6.27. UC.e

```
indexing
    description: "System's root class"
    note: "Initial version automatically generated"

class
    UC

inherit
    ESDL_APPLICATION

    UC_SHARED_ITEMS

create

    make
            -- Creation procedure.


feature -- Initialization

    make
            -- Creation procedure.


end -- class UC
```

3.6.28. UC_LOAD_PUZZLE.e

```
indexing
    description: "load the state of a puzzle saved before in a .puz
     file."
    author: "Markus Neidhart and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_LOAD_PUZZLE

inherit
```

```
    SHARED_QUAD_CUTTER

create
    make


feature -- creation

    make (a_filename: STRING) is
            -- load a .puz file and return a puzzle_start_window
        require
            a_filename_not_Void: a_filename /= Void


feature  -- Implementation

    read_file: PUZZLE_START_WINDOW is
            -- read 'file' in and create a puzzle_start_window


feature {NONE} -- Access

    file: PLAIN_TEXT_FILE


invariant
    file_not_Void: file /= Void


end -- class UC_SAVE_PUZZLE
```

3.6.29. UC_SAVE_PUZZLE.e

```
indexing
    description: "save the state of the current puzzle to an .puz
    file. later, you can load this file and resume the game"
    author: "Markus Neidhart and UC Team"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_SAVE_PUZZLE

inherit
    SHARED_QUAD_CUTTER

create
    make


feature -- creation

    make (a_filename: STRING; situation: PUZZLE_START_WINDOW) is
            -- make an .puz file from 'a_filename'
        require
```

```
            a_filename_not_Void_and_not_empty: a_filename /= Void
            and then not a_filename.is_empty
            situation_not_Void: situation /= Void


feature {NONE} -- Implementation

    save_information (name: STRING; sit: PUZZLE_START_WINDOW) is
            -- store the information in the puz file
        require
            name_not_Void_and_not_empty: name /= Void and then not
            name.is_empty
            sit_not_Void: sit /= Void


feature {NONE} -- Access

    file: PLAIN_TEXT_FILE


end -- class UC_SAVE_PUZZLE
```

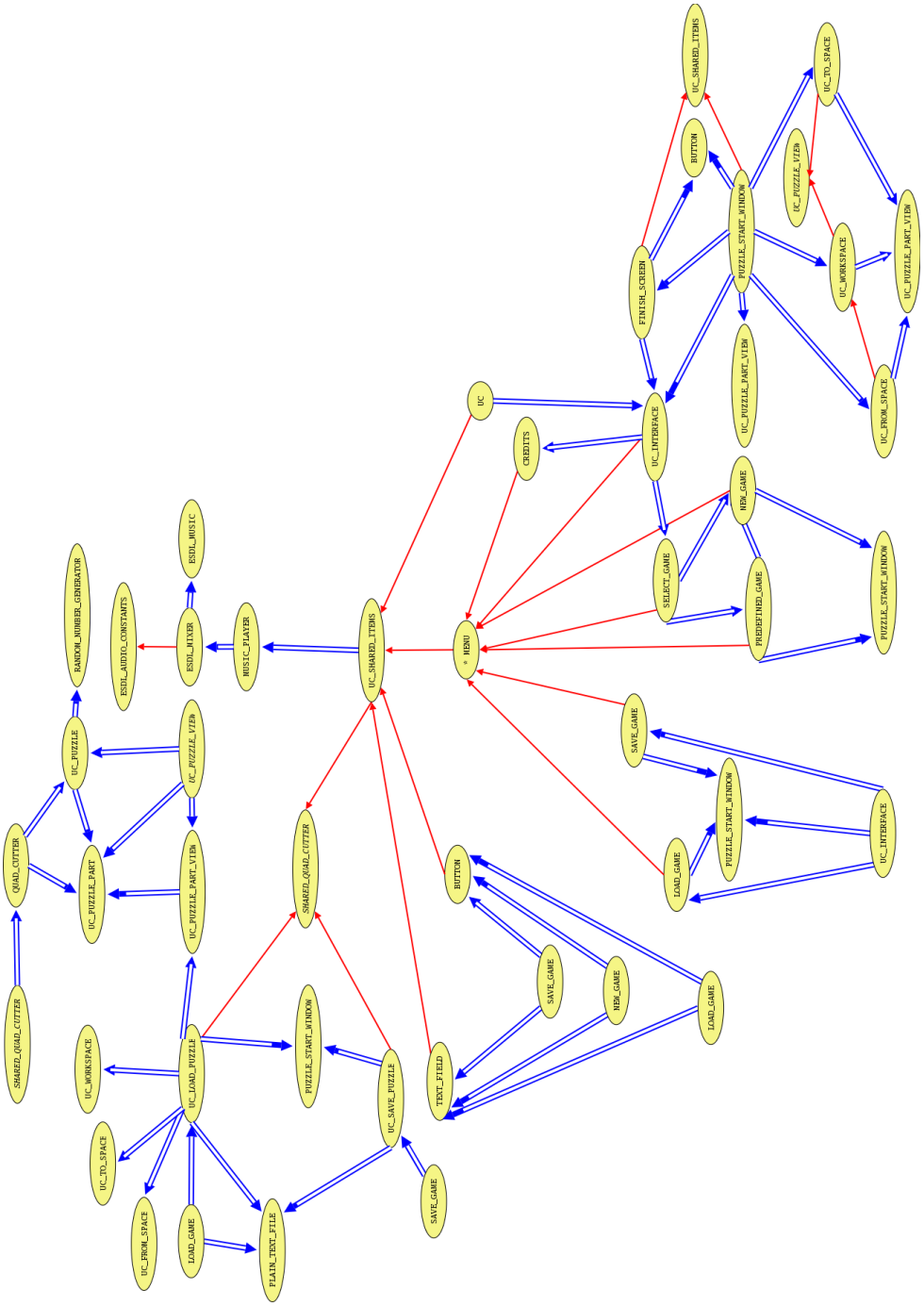
3.6.30. UC_SHARED_ITEMS.e

```
indexing
    description: " Shared access to functions used by UC, extends
    the class ESDL_SHARED_BITMAP_FACTORY"
    author: "Markus Neidhart, nemarkus@student.ethz.ch"
    date: "$Date: 2005/06/20 13:51:31 $"
    revision: "$Revision: 1.3 $"

class
    UC_SHARED_ITEMS

inherit
    ESDL_SHARED_BITMAP_FACTORY
        export
            {NONE} all
        undefine
            default_create
        end

    SHARED_QUAD_CUTTER
        export
            {NONE} all
        end


feature -- Singleton Access

    player: MUSIC_PLAYER is
        -- Sound object which plays sound
    once
        create Result.make
    ensure
```

```
            sound_not_void: Result /= Void
    end


feature -- Colors

        uc_background_color: ESDL_COLOR is
                -- background color
            once
                Result := create {ESDL_COLOR}.make_with_rgb
                (255,204,0)
            end

        uc_light_color: ESDL_COLOR is
                -- light color
            once
                Result := create {ESDL_COLOR}.make_with_rgb
                (255,239,186)
            end

        uc_default_button_active_color: ESDL_COLOR is
                -- default active button color
            once
                Result := create {ESDL_COLOR}.make_with_rgb
                (194,40,0)
            end

        uc_default_button_normal_color: ESDL_COLOR is
                -- default normal button color
            once
                Result := create {ESDL_COLOR}.make_with_rgb
                (255,204,0)
            end

        uc_default_button_prelight_color: ESDL_COLOR is
                -- default prelight button color
            once
                Result := create {ESDL_COLOR}.make_with_rgb
                (255,53,0)
            end

        uc_default_button_highlight_color: ESDL_COLOR is
                -- default highlight button color
            once
                Result := create {ESDL_COLOR}.make_with_rgb
                (255,239,186)
            end

        uc_default_button_border_color: ESDL_COLOR is
                -- default button-border color
            once
                Result := create {ESDL_COLOR}.make_black
            end


feature -- font
        uc_gui_elements_default_font : ESDL_FONT is
                -- default font for the gui_elements
```

```
            once
                bitmap_factory.create_bitmap_from_image ("./image/
                default_font.png")
                check
                    todo_proper_error_handling: bitmap_factory.
                    last_bitmap /= Void
                end
                Result := create {ESDL_BITMAPFONT}.make (
                bitmap_factory.last_bitmap)
            end


feature -- Constants

        width: INTEGER is 1024
            -- The width of the screen

        height: INTEGER is 768
            -- The height of the screen

        resolution: INTEGER is 32
            -- The resolution of the scene


end
```

3.7. BON-Diagram



A. Appendix

A.1. Vergleich zum SRS

In diesem Abschnitt vergleichen wir die Ziele, die wir uns im SRS gesteckt haben, mit dem Endprodukt. Es folgt eine Liste mit Punkten, die genau so umgesetzt wurden, wie sie im SRS standen. Danach folgt eine Liste mit Punkten, die überhaupt nicht, oder nicht so wie im SRS vorgesehen, umgesetzt wurden.

A.1.1. erreichte Punkte

Benutzeroberfläche

Arbeitsflächen: Die Arbeitsflächen wurden wie im SRS vorgesehen umgesetzt. Der Arbeitsfläche **AA** entspricht der Workspace 0. Leider kamen wir nichtmehr dazu eine Suchfunktion zu implementieren und daher macht es auch keinen Sinn, dass der Spieler die Workspaces selbst benennen kann. Infolge Platzmangels (zuwenig Platz für die Buttons) haben wir die Anzahl der Workspaces auf drei beschränkt.

Spielfläche: Die Spielfläche wurde wie im SRS vorgesehen umgesetzt. Man kann ein Puzzle-Teil aus einem Workspace in die Spielfläche ziehen. Falls das Puzzle-Teil am richtigen Ort ist, bleibt es dort liegen, ansonsten springt es zurück in den Workspace.

Gamemenu: Das Gamemenu beinhaltet die Funktionen, welche im SRS vorgesehen waren. Zudem haben wir noch vier weitere Funktionen hinzugefügt. Die "neuen" Funktionen sind Resume Game, Load Game, Save Game, Credits. Die Funktion Resume Game bewirkt, dass man zurück ins Spiel gelangt, aus dem heraus man mit der Taste ESC das Menu aufgerufen hat. Die Funktion Load Game ermöglicht es ein zuvor gespeichertes Puzzle-Spiel zu laden und daran weiterzuspielen. Die Funktion Save Game ermöglicht es das laufende Puzzle-Spiel zu speichern. Die Funktion Credits wechselt zu einem Bildschirm, in dem die Namen der Entwickler des Spiels, sowie Danksagungen an verschiedene Personen die in irgendeiner Weise an der Produktion des Spiels beteiligt waren, angezeigt werden. Wir mussten feststellen, dass es fu"r die Funktion Options keine Verwendung im Gamemenu gibt, da es von der Userseite her keine Einstellungen zu machen gibt.

Funktionen

Speichern / Laden: Die Funktionen Speichern und Laden wurden, bis auf das Schnellspeichern mittels drücken der Tasten Ctrl + S während des spielens, wie im SRS vorgesehen umgesetzt.

Teile drehen: Die Funktion Teile drehen zu können wurde wie im SRS beschreiben umgesetzt.

Maussteuerung: Wie im SRS vorgesehen ist das gesamte Spiel mit der Maus steuerbar.

Hintergrundmusik: Abgesehen davon, dass der Spieler seine Musik nicht im Menu unter Optionen angeben kann, haben wir die Idee der Hintergrundmusik welche während dem Spielen ertönt wie im SRS beschrieben umgesetzt. Zusätzlich zum im SRS beschriebenen Button zum ein und ausschalten der Musik, haben wir noch eine Playersteuerung eingebaut, die entweder mit der Tastatur oder mit der Maus bedient werden kann.

Hilfefunktion: Die Hilfefunktion wurde nicht exakt wie im SRS vorgesehen umgesetzt, ist aber immerhin vorhanden. Es gibt zwei Abweichungen vom SRS im Zusammenhang mit dieser Funktion. Erstens ist die Funktion nicht auf eine bestimmte Anzahl Benutzungen beschränkt. Zweitens wird diese Funktion mit der Taste c (anstatt Ctrl + h) oder dem entsprechenden Button ausgelöst.

Spielmodis

Quadrat-Modus Den Quadrat-Modus haben wir wie im SRS beschriben umgesetzt. Der Spieler kann selbst angeben in wieviele quadratische Teile das Bild zerschnitten werden soll. Zusätzlich kann der Spieler auch selbst das Bild wählen welches er zerschneiden und zusammensetzen möchte.

A.1.2. nicht erreichte Punkte

Benutzeroberfläche

Menubar: Das einzige Element der Menubar, das wir umgesetzt haben ist der Button zum aus- bzw. einschalten der Hintergrundmusik. Da wir keine Suchfunktion implementiert haben, ist die Menubar überflüssig geworden.

Hauptarbeitsfläche: Etwas wie die Hauptarbeitsfläche gibt es in unserer Implementation nicht, da diese ohne Suchfunktion keinen Nutzen hat.

Funktionen

Scrollfunktion: Wir haben keine Scrollfunktion implementiert, da wir keine Zeit mehr hatten. Falls nun zuviele Puzzle-Teile vorhanden sind um alle im Workspace anzuzeigen werden sie über den Workspce hinaus gezeichnet.

Suchfunktion: Wir haben keine Suchfunktion implementiert, da wir keine Zeit dazu hatten. Zudem ist die Suchfunktion für quadratische Puzzle-Teile nicht so wichtig wie für Puzzle-Teile mit Köpfen.

Spielmodis

ETH-Modus: Der ETH-Modus so wie er im SRS beschrieben wurde existiert in unserer Implementation nicht. Es gibt verschiedene predefined Modi, welche im Zusammenhang mit der 150-Jahr-Feier der ETH stehen.

Kopf-Modus: Der Kopf-Modus ist schwieriger zu implementieren als der Quadrat-Modus. Deshalb haben wir mit der Implementation des Quadrat-Modus begonnen. Für den Kopf-Modus blieb uns keine Zeit mehr.

A.1.3. Fazit

Dass wir die Spezifikationen aus dem SRS nicht vollständig umzusetzten vermochten, liegt einerseits sicherlich daran, dass wir beim Schreiben des SRS noch keine Ahnung

von ESDL hatten. Andererseits ist es unmöglich alle Schwierigkeiten (programmiertechnischer oder designspezifischer Art) vorherzusehen. Dadurch haben wir uns beim Schreiben des SRS, was den Aufwand und somit die benötigte Zeit angeht, schlichtweg überschätzt.

A.2. Entwicklung des Projekts

Das Projekt UC - Under Construction wurde im Rahmen der Software Architecture Vorlesung (Bertrand Meyer) im Sommersemester 2005 in einem knapp gehaltenen Zeitfenster entwickelt.

A.2.1. Entwicklungsumgebung

Als Entwicklungsumgebung wurde Eiffel55 und ESDL verwendet (siehe Installation). Unter Linux funktionierte zwar Eiffel korrekt, jedoch konnten wir ESDL (nach langem Basteln) immer noch nicht zum Laufen bringen. EWG und GOBO konnten problemlos installiert werden. Bei SDL mussten auch diverse Libraries hinzugefügt werden (siehe ESDL developer.txt). Beim kompilieren der ESDL libraries traten dann aber Fehler auf, die ich nicht beseitigen konnte. Deshalb entwickelten wir ausschliesslich unter Windows. Es wäre aber theoretisch möglich ESDL unter Linux zu installieren (das hauptsächliche Problem lag, so glaube ich, dass die Umgebungsvariablen nicht korrekt gesetzt wurden, was unterdessen aber wahrscheinlich klar geworden ist).

Damit wir gemeinsam an unserem Projekt arbeiten konnten, benutzten wir das in der Vorlesung vorgeschlagene CVS System. Teilweise direkt in der Konsole ausgeführt, andererseits gibt es auf der Entwicklungsplattform Eclipse auch eine nette CVS Funkton, mit der man das Projekt verwalten konnte. Zum Teil kamen auch andere graphische Front-Ends zum Einsatz.

A.2.2. Start

Nachdem wir unsere Idee ein wenig diskutiert hatten, entwickelten wir nach Vorgabe aus der Vorlesung unser SRS (Software Requirement Specifications, siehe srs.pdf). Nachdem die Installation von ESDL geklappt hatte (Windows) begannen wir mit dem Arbeiten an unserem Projekt. Ein von uns erstelltes BON-Diagram (uc_bon_diagram.png) sollte uns graphisch aufzeigen wie in etwa welche Klassen benötigt werden und wie diese Klassen aufgebaut sind.

Danach begannen wir einzeln an gewissen Teilen zu arbeiten.

A.2.3. Arbeitsverteilung

Eine genaue Arbeitsverteilung gab es zunächst nicht. Jeder arbeitete gerade an einem Teil, der ihm Spass machte. Derjenige war dann aber auch dafür verantwortlich dass sein Teil gut in das ganze System passte und funktionierte. Irgendwie ging dieses Schema bis jetzt gut auf. Alle haben viel am Projekt gearbeitet, jeder hat eine Ahnung was der andere gemacht hat und was dessen Klassen so grob für Funktionen bereitstellen.

A.2.4. Probleme

Im Laufe des Projekts traten auch einige Probleme auf. Bei der Diskussion, wie wir unser Puzzle programmieren stiessen wir schon auf einige Probleme. Wir wussten nicht

wie es möglich wäre ein Fancy Puzzle zu machen (siehe Weiterentwicklungsmöglichkeiten - Fancy Puzzle Part). Wir beschlossen zuerst unser Projekt nur für eckige Teile zu machen. Wenn Zeit bleibt kann dann immer noch erweitert werden. Ebenfalls mussten für unser Projekt Buttons und Texteingabefelder entwickelt werden, um die Kommunikation zwischen User und Programm nicht nur auf Tasten zu beschränken und damit der User etwas eingeben kann (Textfeld), zum Beispiel das Bild, das er für ein neues Puzzle erstellen will.

Einige Probleme hatten wir auch mit dem CVS. Manchmal kam es vor, dass mehrere Personen an derselben Klasse arbeiteten und es deshalb Konflikte beim commit gab. Dies versuchten wir zu verhindern, indem wir immer wieder besprachen, wer gerade was macht. Nach einem Update im CVS wurde dann gerade ein Mail an das ganze Team geschickt. Das CVS hatte auch diverse Bilder "verschrottet". Beim ersten mal hochladen geht nichts schief. Wenn man jedoch ein Bild ändert und dieses commitet, so wird irgendwie von CVS was ins Bild geschrieben und das Bild ist dann kaputt. Also mussten wir immer das Bild löschen und dann wieder hinzufügen. Während dem Arbeiten mit dem CVS haben wir festgestellt, dass man Dateien wie Bilder oder PDF-Dateien als binaries kennzeichnen kann. Solche Dateien werden vom CVS dann nicht "abgeändert". Um Dateien als binaries zu kennzeichnen mussten wir jedoch ausschliesslich das CVS-Plugin von Eclipse benutzen, da wir nicht herausgefunden haben wie man dies in der Kommandozeilen-version von CVS tut.

Beim Programmieren mussten wir auch kleine Änderungen an ESDL vornehmen (siehe diverse hacks), um die gewünschten Funktionen zu haben.

Im grossen und ganzen kann man aber sagen, dass wir gut zurechtkamen.

A.3. Testing

Für das Testen bleibt meistens nicht mehr allzu viel Zeit übrig. Es könnte durchaus sein, dass unser Programm ab und zu mal einen Schönheitsfehler beinhalten kann.

Allerdings wurden die geschriebenen Klassen laufend wieder getestet und sollten stabil laufen.

A.3.1. Bekannte Bugs

Diese Liste beinhaltet Fehler, die nach dem Beenden der Programmierphase noch entdeckt wurden, oder Fehler die wir schon während dem Programmieren kannten aber nicht beheben konnten.

- Während dem spielen werden Teilchen die gedreht wurden durch den Cheaterbutton nicht richtig postitioniert.
- Beim Drehen eines Teilchens entsteht ein schmaler schwarzer Rand um das gedrehte Teilchen herum.
- Wenn ein Spiel gespeichert wird und schon ein gespeichertes file existiert crasht das game.
- ...

A.4. Weiterentwicklungsmöglichkeiten

A.4.1. Sound

Aus unserer anfänglichen Idee, dass der Spieler im Hintergrund Musik abspielen kann, ist inzwischen ein ganzer Music Player entstanden. Jener kann aber nur .ogg files abspielen. Man könnte nun den Player erweitern, auch mp3s oder andere Formate, abzuspielen. Dies kann aber zurückgehen bis auf die SDL libraries (siehe ESDL_MIXER, ESDL_AUDIO_CONSTANTS). Wir hatten zuwenig Zeit um uns länger damit zu beschäftigen.

A.4.2. Modus

Ursprünglich hatten wir im Sinn mehrere Modi zu machen. Wenn der Spieler ein neues Spiel kreiert, dann hätte er die Möglichkeit auszuwählen. Der Spieler kann zwar jetzt keine Modi auswählen, jedoch kann er mit der Einstellung der Anzahl Teilchen die Schwierigkeit bestimmen. Die vordefinierten Einstellungen dienen als Ersatz dieser Funktion.

A.4.3. Fancy Puzzle Part

Im jetztigen UC gibt es nur viereckige Puzzle Teilchen. Man könnte aber auch solche mit Köpfen und Löcher implementieren. Dazu müsste man zusätzlich in einem Puzzle Teilen beim zerschneiden speichern, wo Köpfe, wo Löcher sind. Um so ein Fancy Teil zu verwirklichen könnte man ein Puzzle Teil grösser machen und rundherum eine als transparent definierte Farbe nehmen. Oder man lädt das Puzzle Teil anders und zeichnet vom benachbarten Puzzle Teil noch den Kopf dran. Für diese verschiedenen Implementationen (quad, fancy) hatten wir in unserem SRS Klassen vorgesehen, die deferred features besitzen.

A.4.4. Join von mehreren Puzzle Teilen

Uns kam auch die Idee, das man nicht nur auf der Spielfläche die Puzzle Teile zusammensetzen könnte, sondern auch gerade in einem workspace. Würden 2 Teile zusammenpassen so würden diese einfach zu einem neuen Teil verschmelzen.

A.4.5. Scrollfunktionen

Wenn man sehr viele Teile hat oder grosse Bilder, dann gibt es ein Problem mit der Darstellung. Es können unmöglich alle Puzzleteile auf dem Bild dargestellt werden. Eventuell hat auch das Puzzle kein Platz auf dem Screen. Man könnte deshalb für die einzelnen Workspaces das Scrollen implementieren. Fährt man an einen Rand des workspaces (unten, oben) so bewegt sich der workspace in die gewünschte Richtung soweit möglich.

A.4.6. Zoom

Im Zusammenhang mit den obigen Scrollfunktionen könnte man auch Zoom hinzufügen, um grosse Puzzlebilder ganz anzeigen zu können.

A.4.7. Filter / Search

Man könnte einen Filter oder eine Suche einbauen, um die Puzzle Teile zu sortieren. Man könnte nach Rand oder Ecken suchen. (bei Fancy Teilen auch nach Köpfen oder

Löcher). Das Resultat der Suche könnte man in einem neuen workspace darstellen, oder man könnte den Filter direkt auf den workspace anwenden um so die gewünschten Teile zu erhalten. Ebenso wäre eine Sortierung nach Farbe vorstellbar. Dies wäre jedoch schwierig zu realisieren. Es müsste irgendwie einen Farbwert aus allen Farbwerten berechnet werden. Die Zuteilung könnte eventuell nicht dem entsprechen, was der Spieler von der Zuteilungsfunktion erwartet.

A.5. Wiederverwendbarkeit

A.5.1. Music-Player

Die Klasse music_player.e bietet einen kompletten Player für Musik. Auf Grund der fehlenden Unterstützung durch ESDL spielt dieser im Moment ausschliesslich .ogg-Files. Sobald die anderen Formate aber von der Library unterstützt werden, kann der Player auch für diese Formate verwendet werden.

Wie man ein Gui für den Player implementieren könnte, sieht man im PSW, das einige Buttons für die Steuerung des Players bereitstellt.

A.5.2. Gui-Elemente

Die Klassen button.e und text_field.e bieten sehr allgemeine Gui-Elemente an, die problemlos wiederverwendet werden können und vielseitige Einstellungsmöglichkeiten bieten.

A.5.3. Zufalls-Generator

Die Klasse random_number_generator.e stellt allgemeine Zufallsfunktionen zur Verfügung, welche den Umgang mit dem von der Eiffel-Library zur Verfügung gestellten Zufallsgenerator vereinfachen.

A.5.4. Mausunterstützung im Menu

Weiterhin wurde die Klasse menu.e vom UC-Team mit Maussteuerung erweitert.

A.5.5. Behandlung von Key- und anderen Events

Die Struktur der Agents zur Behandlung der Events, insbesondere der Key-Events kann für andere Anwendungen einfach angepasst werden.

A.5.6. Weiteres

Siehe auch *spezielle Klassen*.

A.6. Special Thanks to

- Benno Baugartner for a cool Assistance
- Sandro Blum for the sound at the presentation (visit: sound.selenic.net)