Semestrální práce

z předmětů testování softwaru

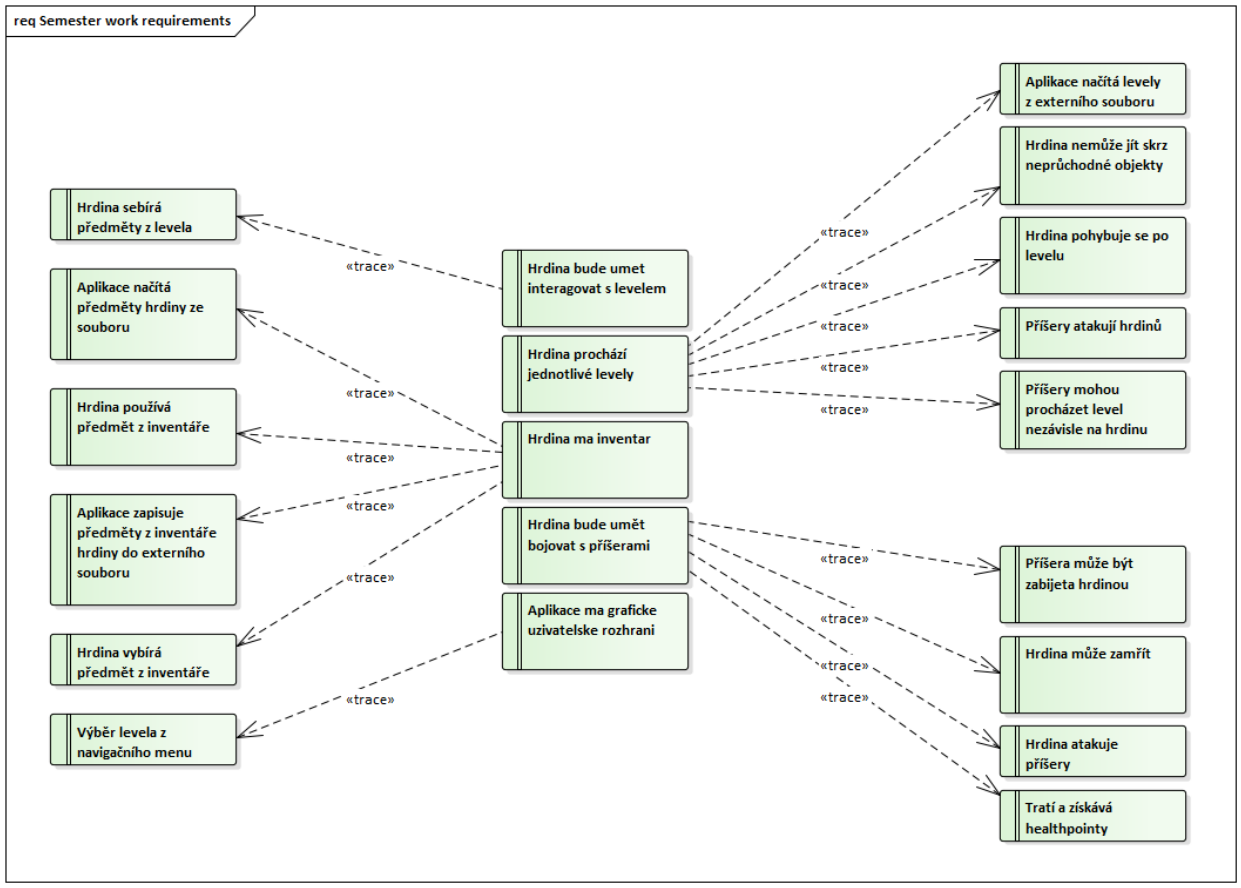
projekt pro testování: GameEngine

Ivan Shalaev
FEL ČVUT 2021
shalaiva@fel.cvut.cz

Návrh testovací strategie

Popis projektu pro testování

Projektem je jednoduchý herní engine, který jsem vytvořil v rámcích semestrální práce z předmětu PJV. Ze specifikací požadavku semestralky z PJV můžeme vyjádřit systémové požadavky projektu rozdělením původních požadavků na dílčí části.

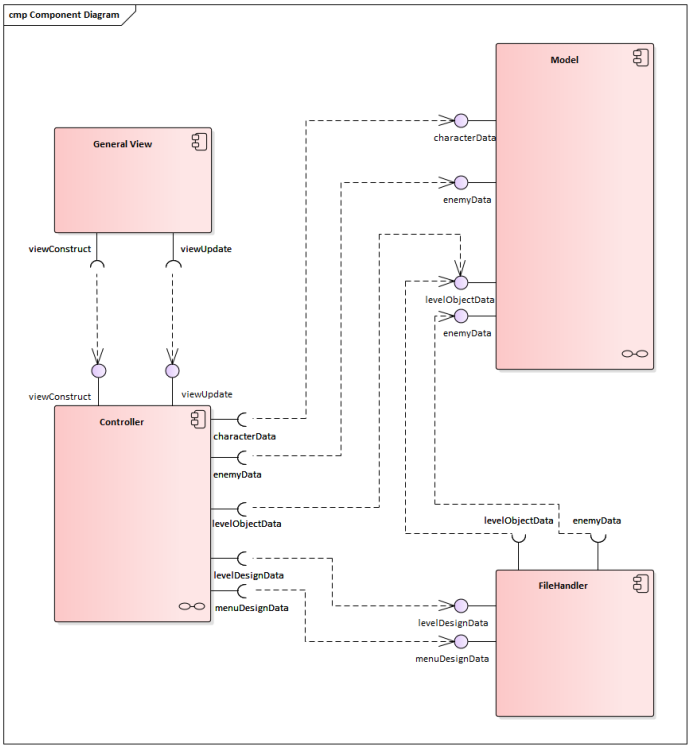


Navíc jsem ještě implementoval jednoduchý generátor objektů na úrovni z textovou mapy. Dále jsou screenshoty mapy objektu a tilu zemi

```
@-------------------------#
|.ab.abo.ab.abqQ..ab.....abH|
|.cd.cdabcd.cd.ab.cd.ab..cd.|
|ab.H..cd..ab..cd....cd.....|
|cd........cd.....ab........|
|..ab..o......o...cd....S...|
|..cd......S.................|
}-----#.....................|
|GGGGGv/.....................|
|12ES........................|
|34..........................|
|GGGGG^/.....................|
}-----$....abab.............|
|qQo....ab.cdcd.....S.......|
|abqQab.cd.ab.o..............|
|cd.ocd....cd.......abab....|
|ab.................cdcd..H.|
|cdababqQ..........ab..abab.|
|qQcdcdqQ..........cd..cdcd.|
%-------------------------$
```

```
****************************
****************************
****************************
**************<-----i*******
********************|*******
********************|*******
********************|*******
^^^^^^^^^^^^^^^^#****|*******
+++++++++++++++)****|*******
+++++++++++++++]----X-------
____;++++++++++)****|*******
****(++++++++++)****|*******
****%__________$****|*******
********************|*******
******************F---X-------
*****************|****v*******
*****************|************
***********F----J************
***********|*****************
***********|*<-->************
```

Testovaci strategie



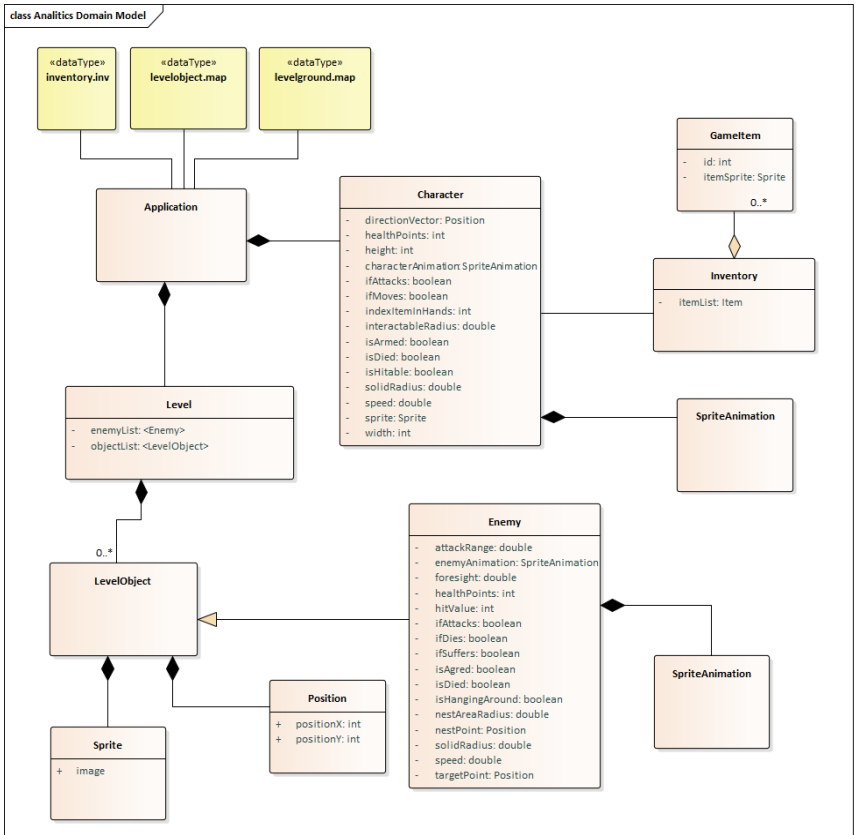
Samotný program byl sestaven podle architektury MVC a podle ní bych rozděloval celý program na komponenty.

Tím pádem máme **části**:

- ❏ Modul správce úrovně
- ❏ Modul správce hrdiny
- ❏ Modul správce nepřátele
- ❏ Modul zobrazení
- ❏ Modul správce souborů

A **procesy**:

- ❏ Načtení soubora
- ❏ Zápis do soubora
- ❏ Zobrazení (grafické)
- ❏ Ošetření události hrdiny
- ❏ Ošetření události nepřátele



Doplněná tabulka tříd rizika

Characteristic: Functionality								
Proces	Podproces	Požadavek	Mozne poskozeni	Vysvětlení možného poškození	Část systému	Pravdepodobnost selhani	Vysvětlení pravděpodobnost selhání	Třída rizika
Zobrazení	-	Aplikace má grafické uživatelské rozhraní	High	GUI ve vře je primárním aspektem pro dojem spotřebitelů	Modul zobrazení	Low	Podle myšlenky ten modul jen zobrazuje data, většinou chyby s zobrazením je chybami ostatních modulů	B
Načtení soubora	Načtení levelu	Aplikace načítá levely z externího souboru		Velký riziko spadnutí klesání celé hry	Modul správce souborů	Medium	Je dost jednoduchý, většina výjimek ošetřena	
	Načtení inventáře	Aplikace načítá předměty hrdiny ze souboru	Medium	Může působit neprůchodné herní situace				
Zápis do soubora	-	Aplikace zapisuje předměty z inventáře hrdiny do externího souboru		Může dojít ke ztrátě dat				
Ošetření události nepřátele	Pronásledování cíle	Příšery atakují hrdinů	Low	Většinou neruší procházení hrou	Modul správce nepřátele	High	Modul obsahuje hodně funkcí komplikovaných výpočtů a herních mechanik, běží ve vlastním vlákně v cyklu tj. má velkou frekvenci používání	C
	Nezávislé uměle chování	Příšery mohou procházet level nezávisle na hrdinu						
	Sledování okamžitého stavu	Příšera může být zabijeta hrdinou						
Ošetření události hrdiny	Interagováni s předměty	Hrdina používá předmět z inventáře			Modul spravce hrdiny			
	Ošetření událostí klávesnice	Hrdina vybírá předmět z inventáře						
	Ošetření událostí klávesnice	Hrdina pohybuje se po levelu	High	Základní mechanika hry, má velký vliv na gameplay. Může působit neočekávané herní situace				A
	Výpočet kolizí	Hrdina nemůže jít skrz neprůchodné objekty						
	Interagováni s předměty	Získává healthpointy	Low	Většinou neruší procházení hrou				C
	Sledování okamžitého stavu	Hrdina může zamřít						
	Interagováni s urovnem	Hrdina sebírá předměty z levela						
		Hrdina atakuje příšery	Medium	Základní mechanika hry				B

		Pravděpodobnost selhání		
		High	Medium	Low
Možné poškození v případě selhání	High	A	B	B
	Medium	B	B	C
	Low	C	C	C

Určení intenzity

V rámcích této semestralky budeme uvažovat Unit testy a Procesní testy. V projektu GameEngine nejsou poskytnuté souvislostí s jinými systémy a proto, v souladu s definicí, Integrační testy provázet nebudeme.

Bezchybná funkcionalita (intenzita)							
Část systému / Funkce	Třída rizika	Revize	Vývojářské testy		Integrační testy	Systémové testy	UAT
			Unit testy	Procesní testy			
Modul zobrazení	B	Ano	-	-	-	-	Medium
Modul správce souborů							
levelInit	B	-	High	Low	-	Low	-
groundTilesInit							
saveInventroy							
Modul správce nepřátele							
changeHealthPoints	C	-	Low	Low	-	Low	-
force							
seePoint							
enemyGetsHit							
hangAround							
move							
attack							
getCollisionPoints							
Modul správce hrdiny							
move	A	-	High	Medium	-	Medium	Low
setDirection							
getCollisionPointList							
keyPress							
keyUnpress							
pressedMoveKeysQuantity							
attack	B						
changeItemInHand	C		Medium				
interactWithItem							
interactWithObject							
heal							
characterGetsHit							
putItem							
removeFromInventory							

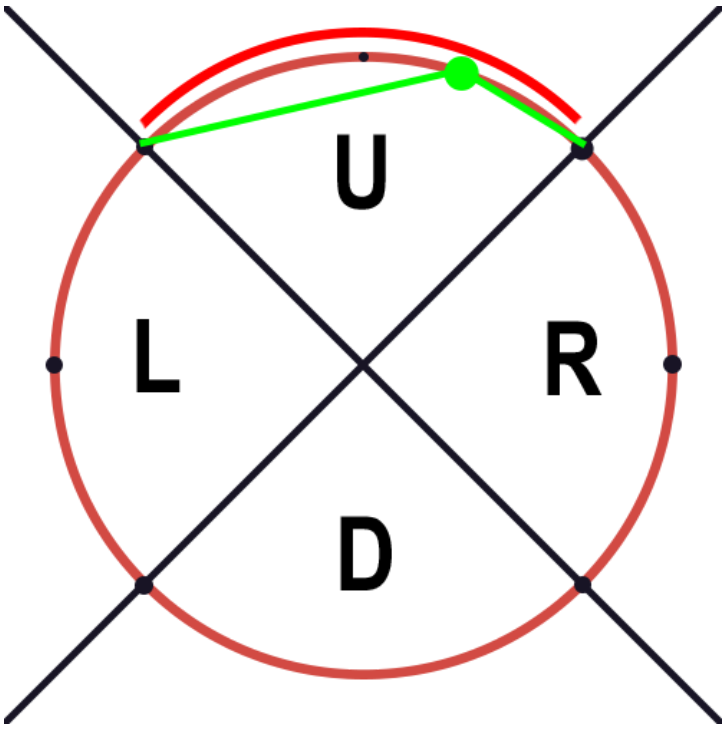
CRUD Matice

Proces	Entity				
	Character	Enemy	GameItem	LevelObject	Inventory
Zobrazení	R	R	R	R	R
Načtení levelu	-	R	-	R	-
Načtení inventáře	U	-	-	-	-
Zápis do soubora	R	-	-	-	-
Pronásledování cíle	R	U	-	R	-
Nezávislé uměle chování	-	R/U	-	-	-
Sledování okamzitoho stavu	R	R	-	-	-
Interagováni s předměty	R/U	-	R/D	-	R/U
Ošetření událostí klávesnice	U	-	-	-	-
Výpočet kolizí	R/U	R/U	-	R	-
Interagováni s urovnem	R	R/U	-	R/U	R/U

Testovací scénáře

Třídy ekvivalence pro funkci setDirection *EnemyController*

S dokumentací víme: tato funkce vrací prostou stringovou reprezentaci směrového vektoru nepřítele. Tato funkce počítá délky úseček od konce jednotkového směrového vektoru do dvou diagonálních bodu (např. do bodu UP-LEFT a UP-RIGHT), pak skládá ty hodnoty a porovnává s délkou ¼ kružnice. Když ten součet je menší, zapisuje do řetězce slovní reprezentaci směru (napr. ÚP).



Tříd ekvivalence bude 4 podle výstupů a každá bude reprezentována intervalem vektoru mezi diagonálními body V dokumentaci není jednoznačně určený mezní podmínky a proto řekneme, že vektory s končí v UP-LEFT a DOWN-LEFT patří do EC "LEFT" a vektory s konci UP-RIGHT a DOWN-RIGHT patří do EC "RIGHT". Jedná se o jednotkových vektorech a o kružnici s r = 1. Z toho určíme vektory, generující každou EC (A = $\begin{pmatrix} cos\,\alpha & -sin\alpha \\ sin\,\alpha & cos\alpha \end{pmatrix}$, v = (jednotkový) vektor s konci v UP-RIGHT) :

1. LEFT = $\{v_L |\, v_L = Av,\ 60 \leq \alpha \leq 120\}$
2. RIGHT = $\{v_R |\, v_R = Av,\ -60 \leq \alpha \leq 0\}$
3. UP = $\{v_U |\, v_U = Av,\ 0 < \alpha < 60\}$
4. DOWN = $\{v_D |\, v_D = Av,\ -120 < \alpha < -60\}$

V programu pro reprezentaci vektorů se používá třída Position mající parametry x a y. Z toho máme:

1. LEFT = $\{x, y |\ -1 \leq x \leq -0.707,\ |y| \leq 0.707\}$
2. RIGHT = $\{x, y |\, 0.707 \leq x \leq 1,\ |y| \leq 0.707\}$
3. UP = $\{x, y |\, |x| \leq 0.707,\ 0.707 \leq y \leq 1\}$
4. DOWN = $\{x, y |\, |x| \leq 0.707,\ -1 \leq y \leq -0.707\}$

Vím, že tato funkce má triviální vstup, proto uměle zkomplikujeme vstupy a řekněme, že kromě směrového vektoru funkce bude přijímat na vstup jese 3 parametry: int a, int b, String č. Doplňková podmínka je, když a > b výstup je c, jinak podle logiky funkci výše. Z toho určíme nové EC:

1. LEFT = $\{v_L, a, b, c | (a > b \cap c = "LEFT") \cup (a \leq b \cap v_L = Av \cap 60 \leq \alpha \leq 120)\}$
2. RIGHT = $\{v_R, a, b, c | (a > b \cap c = "RIGHT") \cup (a \leq b \cap v_R = Av \cap -60 \leq \alpha \leq 0)\}$
3. UP = $\{v_U, a, b, c | (a > b \cap c = "UP") \cup (a \leq b \cap v_U = Av \cap 0 < \alpha < 60)\}$
4. DOWN = $\{v_D, a, b, c | (a > b \cap c = "DOWN") \cup (a \leq b \cap v_D = Av \cap -120 < \alpha < -60)\}$

Určíme vstupní data pokrývající všechny EC:

- $v_L = \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix}$, $v_L = \begin{pmatrix} -0.707 \\ -0.707 \end{pmatrix}$
- $v_R = \begin{pmatrix} 0.707 \\ 0.707 \end{pmatrix}$, $v_R = \begin{pmatrix} 0.707 \\ -0.707 \end{pmatrix}$
- $v_U = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- $v_D = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$
- $c = "LEFT", c = "RIGHT", c = "UP", c = "DOWN"$
- $a = -1, a = 1$
- $b = 0, b = 1$

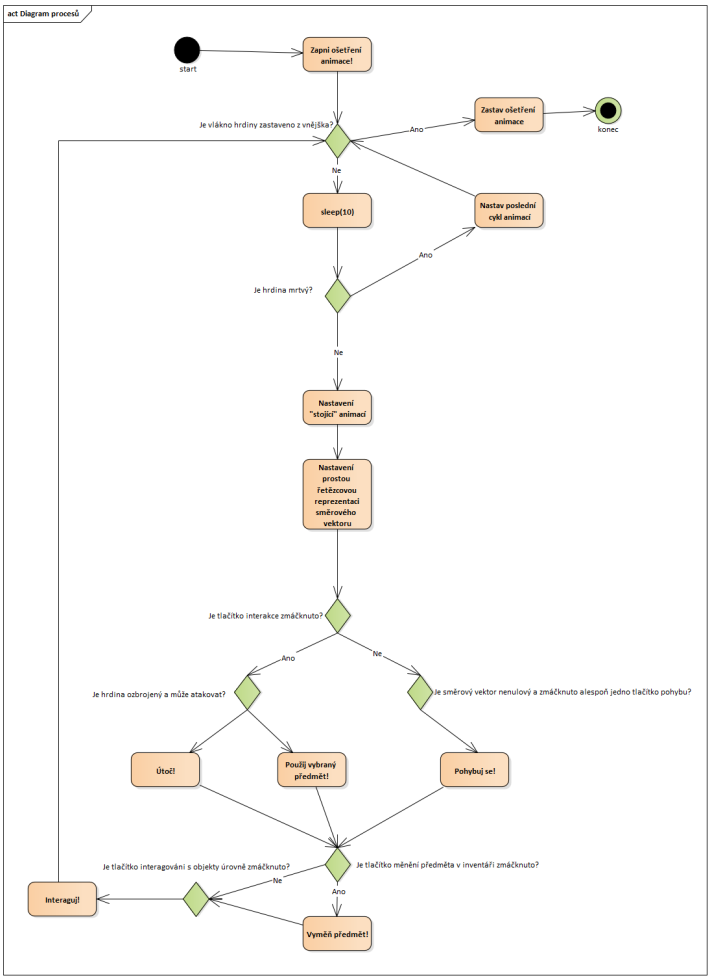
Z toho máme scénáře kde pro ***v*** platí:

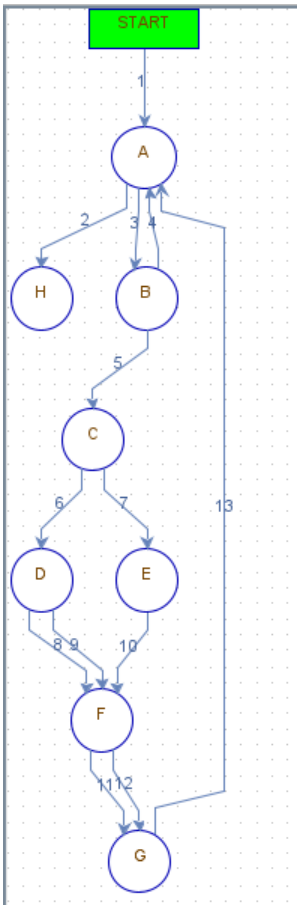
1. $UPLEFT = \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix}$
2. $DOWNLEFT = \begin{pmatrix} -0.707 \\ -0.707 \end{pmatrix}$
3. $UPRIGHT = \begin{pmatrix} 0.707 \\ 0.707 \end{pmatrix}$
4. $DOWNRIGHT = \begin{pmatrix} 0.707 \\ -0.707 \end{pmatrix}$
5. $UP = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
6. $DOWN = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$

ACTS Test Suite Generation: Mon May 24 12:43:26 CEST 2021
'*' represents don't care value
Degree of interaction coverage: 2
Number of parameters: 4
Maximum number of values per parameter: 6
Number of configurations: 24

Test Case No.	a	b	c	v
1	-1	1	LEFT	UPLEFT
2	1	0	RIGHT	UPLEFT
3	-1	0	UP	UPLEFT
4	1	1	DOWN	UPLEFT
5	1	0	LEFT	DOWNLEFT
6	-1	1	RIGHT	DOWNLEFT
7	1	1	UP	DOWNLEFT
8	-1	0	DOWN	DOWNLEFT
9	1	1	LEFT	UPRIGHT
10	-1	0	RIGHT	UPRIGHT
11	-1	1	UP	UPRIGHT
12	1	1	DOWN	UPRIGHT
13	1	1	LEFT	DOWNRIGHT
14	-1	0	RIGHT	DOWNRIGHT
15	-1	0	UP	DOWNRIGHT
16	-1	0	DOWN	DOWNRIGHT
17	1	1	LEFT	UP
18	-1	0	RIGHT	UP
19	-1	1	UP	UP
20	1	0	DOWN	UP
21	1	1	LEFT	DOWN
22	-1	0	RIGHT	DOWN
23	1	1	UP	DOWN
24	-1	1	DOWN	DOWN

Procesní diagram (ošetření události hrdniny)





Test situations 5, TDL= 2, ALG= PCT.

Sub-combinations of edges | Test situations

Node	Sub-combinations of edges
D	6 - 8 6 - 9
C	5 - 6 5 - 7
B	3 - 4 3 - 5
A	1 - 2 1 - 3 4 - 2 4 - 3 13 - 2 13 - 3
G	11 - 13 12 - 13
E	7 - 10
F	8 - 11

Test situations 4, TDL= 2, ALG= PCT.

Sub-combinations of edges | Test situations

No.	Test sequence
1	1 - 2
2	1 - 3 - 4 - 2
3	1 - 3 - 5 - 6 - 8 - 11 - 13 - 3 - 4 - 3 - 5 - 7 - 10 - 11 - 13 - 2
4	1 - 3 - 5 - 6 - 9 - 11 - 13 - 2
5	1 - 3 - 5 - 6 - 8 - 12 - 13 - 2
6	1 - 3 - 5 - 6 - 9 - 12 - 13 - 2
7	1 - 3 - 5 - 7 - 10 - 12 - 13 - 2

Situations: 7, Steps: 54, High: 0, Medium: 0, Low: 54, Unique nodes: 8, Unique edg

Detailní testovací scénář

Parametr	Obsah
ID testu	4
Sekvence	1-3-5-6-9-11-13-2
Název testu	Test ošetření události hrdiny
Hloubka detailů	Střední
Popis testu	Hrdina pohybuje se po úrovni, zmačkne tlačítko interagování s předmětem, interagování s urovnem, interagování s inventářem a zemře
Vstupní podmínky	Živý hrdina má v inventáři předmět (kromě zbroje)
Testovací data	• character.inventory • simpleDir
Očekávaný výsledek	Prázdný inventář, simpleDir=”RIGHT”

Přehled implementací testů

InventoryFileHandlerTest

Test funkci saveInventory, prověřuje správnost zápisu inventáře do souboru (podle doku přepisuje se IDčka v soubor save.inv), používáno mockování pro třídu Inventary.

CharacterControllerTest

3 unit testy s mockováním:

move_directionVectorUP_movesUp() - mockuje se *directionVector* (hrdina posune se ve směru *directionVector*), spustí se *move()*, ověřuje se posun hrdiny podle ho posize.

changeItemInHand_2items_2ndItemInHand() - mockuje se *Inventory*, "vkládá se" do toho mocky *Item* a *Weapon*(dědí se od *Item*), vytváří se hrdina s tímto inventářem (defaultně první předmět je *itemInHand*), spustí se *changeItemInHand()*, ověřuje se, že *itemInHand* je ten *Weapon*.

interactWithItem_2items_2ndItemInHand() - mockuje se *Inventory*, vkládá se" do toho mocky *Item*'y, přičemž jeden předmět je heal, tj. interaktivní (podle IDčka), vytváří se hrdina s tímto inventářem (defaultně první předmět je *itemInHand*), spustí se *interactWithItem()*, ověřuje se, že *itemInHand* je teď ten druhý předmět (první byl použit a smazán)

4 procesni testy:

processTest1() - testuje kombinace metod *setDirection()*, *move()*, *characterGetsHit()*, *interactWithItem()*, po každé metodě se ověřuje správnost výsledků:

setDirection() mění *simpleDir*

move() mění pozice

characterGetsHit() mění HP hrdiny

interactWithItem() v tomto případě mění HP (používá se heal)

processTest2() - testuje kombinace metod *changeItemInHand()*, *attack()*, *characterGetsHit()*, po každé metodě se ověřuje správnost výsledků:

changeItemInHand() - mění *itemInHand*

attack() - mění *attacks* (bool proměnná)

characterGetsHit() - v tomto případě mění *isDied* (bool proměnná)

procesniTest3() - testuje kombinace metod *changeItemInHand()*, *act()*, *changeItemInHand()* x3, *interactWithItem()*, po každé metodě se oveřuje správnost výsledků:

changeItemInHand() - mění *itemInHand*

act() - mění *acts* (bool proměnná)

interactWithItem() - mění *itemInHand*

procesniTest4() - testuje kombinace metod *keyPress()*, *setDirection()*, *move()*, *keyUnpress()*, *setDirection()*, *move()*, po každé metodě se ověřuje správnost výsledků:

keyPress() - nemůžeme ověřit vliv metody, proto ověříme nepřímo po volání následující metody

setDirection() - mění *simpleDir* (v závisnosti od *keyPressed()*)

move() mění pozici

PositionTest

Parametrické unit testy funkcí třídy *Position*. Všechny funkci *Position* mají nekonečně mnoho EC (podle výstupů), proto jsem napsal parametrické testy pokrývající celé/desítkové a kladné/záporné vstupy