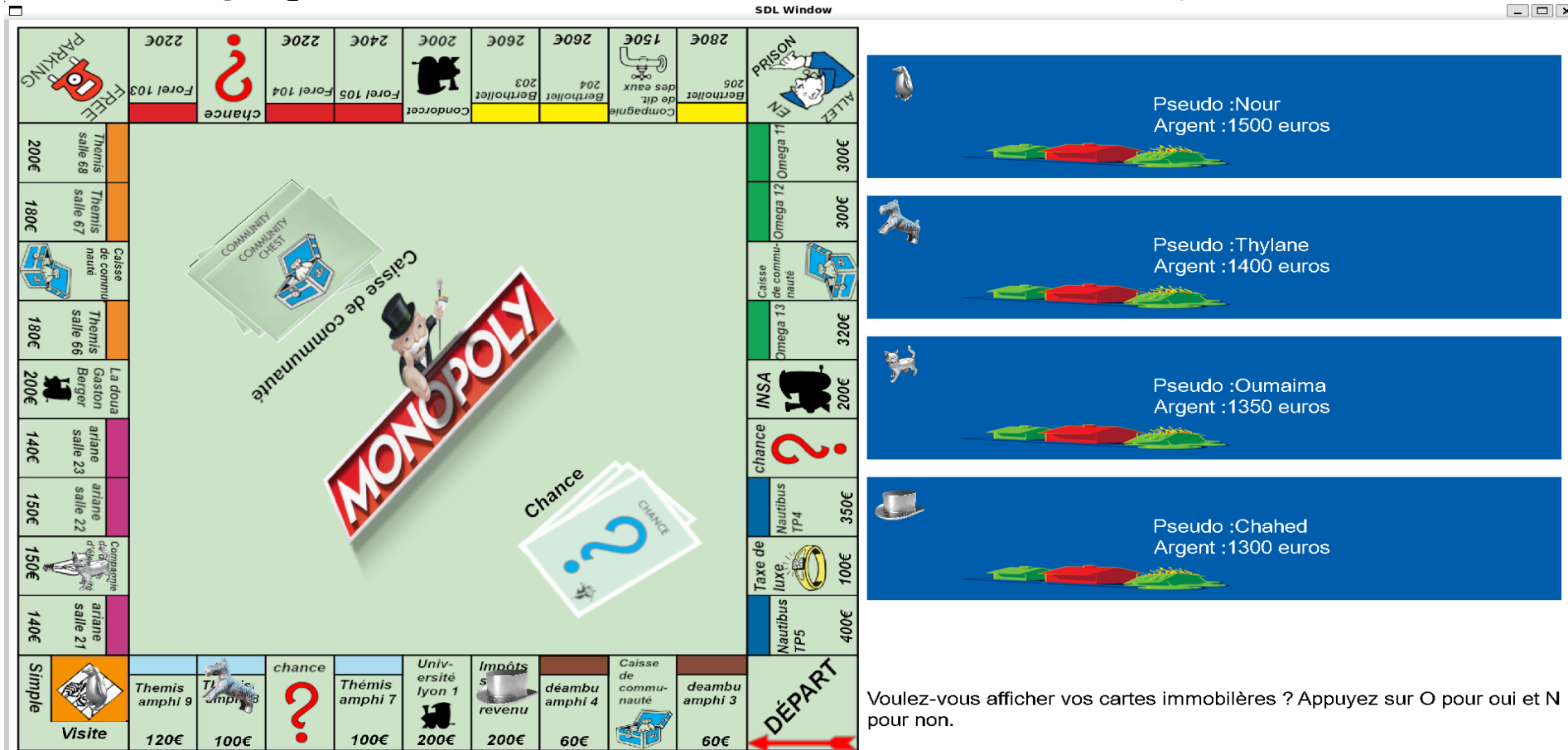


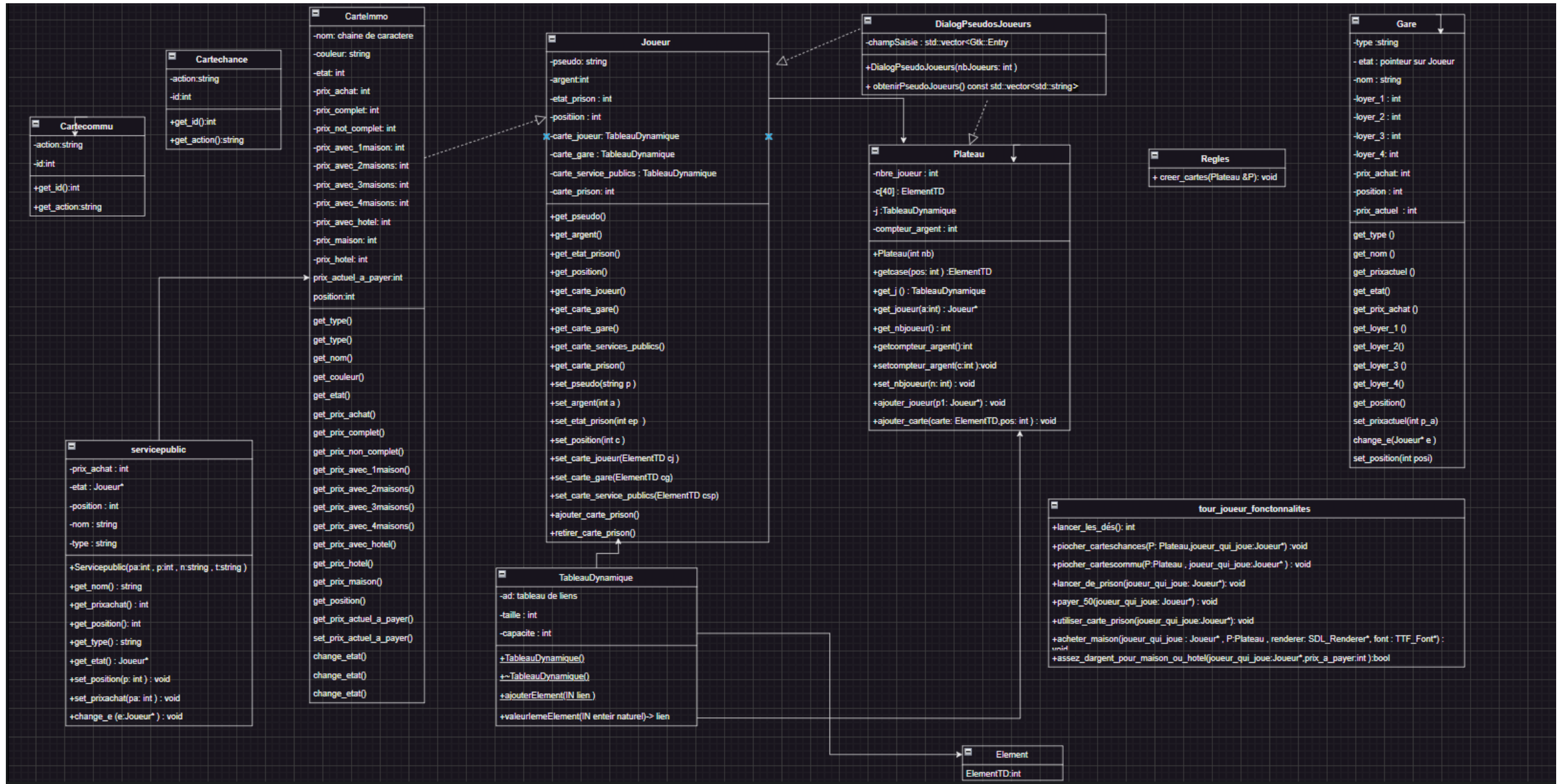
Monopoly Lyon 1 :

Le Monopoly Lyon 1 est un jeu de société où les joueurs achètent, vendent des propriétés et construisent des maisons et des hôtels sur le campus LyonTech-la Doua . L'objectif est de gérer ses finances avec stratégie pour ruiner ses adversaires et être le dernier joueur solvable.

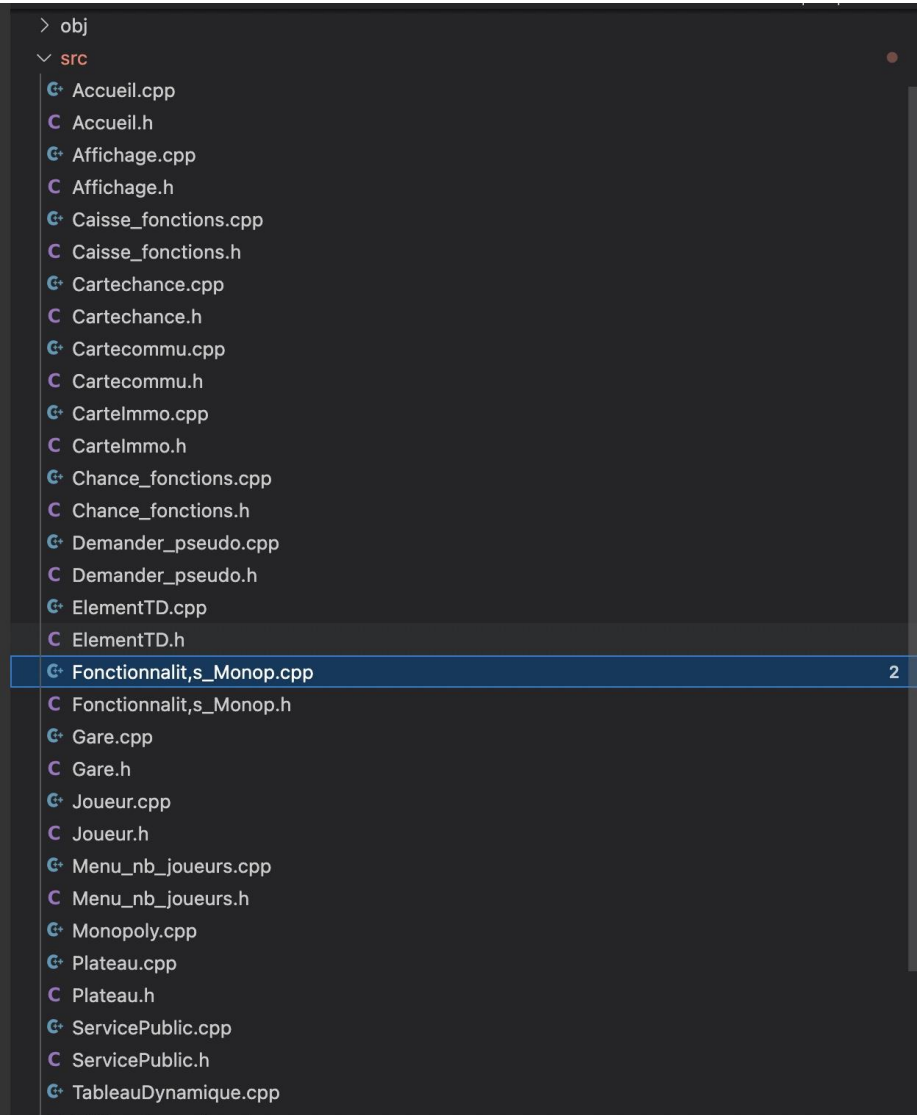


Auteurs: Thylane Rossi , Nour Baananou, Oumaima Tali

Diagrammes des classes



Complexité du jeu ; plusieurs règles, conditions , dépendances :



Plusieurs classes representant les differents éléments du jeu ;
Cartechance, Cartecommune , Carteimmo ,Joueur ...

Exemple de règle complexe : Sortir de prison (code dans la prochaine diapo)

- Concept : si le joueur tombe sur la case 30, il part en prison . Il y'a 3 options pour sortir de prison ; payer 50 euros , utiliser une carte “libéré de prison “ ou obtenir 2 fois de suite un double avec les dés .
- Cet exemple n'est pas compliqué à coder mais il représente la majorité des regles de monopoly ; elles sont longues à coder et il faut prendre en compte toutes les conditions .
- Pour coder notre fonction et pour que le jeu soit fluide , on devait à chaque fois vérifier l'argent et les cartes du joueur et lui proposer de choisir entre les options possibles à lui .

```

76 // Premiere chose, tester si le joueur est en prison
77
78 if (joueur_qui_loue-get_etat_p_rison())<0{
79
80     p_rison = true;
81
82     //Initialisation des choix possibles pour sortir de prison
83
84     bool choix_payer = false;
85     bool choix_lancer = false;
86     bool choix_carte = false;
87     bool choix_fait = false;
88
89     //Début des choix pendant le tour
90
91     SDL_Event event;
92     p_rison = true;
93     while (p_rison)
94     {
95
96         if (SDL_PollEvent(&event))
97         {
98             SDL_PumpEvents(); // Mettre à jour l'état du clavier
99             const Uint8* keystates = SDL_GetKeyboardState(NULL);
100
101             //== le joueur a juste le choix de lancer les dés -> il n'a pas de carte p_rison et pas assez d'argent pour payer 50€ ==>
102
103             if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0)
104             {
105                 afficher_commentaireP, renderer, font, "Vous n'avez qu'un seul choix pour sortir de prison: Lancer les dés, si vous faites un double vous
106                 lancer_de_p_rison(P,joueur_qui_loue, renderer, font);
107                 p_rison = false;
108             }
109
110             //== le joueur a 2 choix : lancer les dés et payer 50€ pour sortir de prison -> il n'a pas de carte p_rison ==>
111
112             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0)
113             {
114                 afficher_commentaire_DemandeP, renderer, font, "Vous avez deux choix pour sortir de prison: Choisissez soit lancer les dés, pour tenter
115                 choix_fait = false;
116                 choix_lancer = false;
117             }
118
119             //== le joueur a 3 choix : lancer les dés, payer 50€ et se faire une carte p_rison -> il n'a pas de carte p_rison ==>
120
121             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0)
122             {
123                 afficher_commentaire_P3, renderer, font, "Vous avez trois choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
124                 choix_fait = false;
125                 choix_lancer = false;
126                 choix_payer = false;
127             }
128
129             //== le joueur a 4 choix : lancer les dés, payer 50€, se faire une carte p_rison et se faire un dé -> il n'a pas de carte p_rison ==>
130
131             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
132             {
133                 afficher_commentaire_P4, renderer, font, "Vous avez quatre choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
134                 choix_fait = false;
135                 choix_lancer = false;
136                 choix_payer = false;
137                 choix_carte = false;
138             }
139
140             //== le joueur a 5 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
141
142             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
143             {
144                 afficher_commentaire_P5, renderer, font, "Vous avez cinq choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
145                 choix_fait = false;
146                 choix_lancer = false;
147                 choix_payer = false;
148                 choix_carte = false;
149                 choix_nb_de = false;
150             }
151
152             //== le joueur a 6 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
153
154             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
155             {
156                 afficher_commentaire_P6, renderer, font, "Vous avez six choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
157                 choix_fait = false;
158                 choix_lancer = false;
159                 choix_payer = false;
160                 choix_carte = false;
161                 choix_nb_de = false;
162             }
163
164             //== le joueur a 7 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
165
166             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
167             {
168                 afficher_commentaire_P7, renderer, font, "Vous avez sept choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
169                 choix_fait = false;
170                 choix_lancer = false;
171                 choix_payer = false;
172                 choix_carte = false;
173                 choix_nb_de = false;
174             }
175
176             //== le joueur a 8 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
177
178             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
179             {
180                 afficher_commentaire_P8, renderer, font, "Vous avez huit choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
181                 choix_fait = false;
182                 choix_lancer = false;
183                 choix_payer = false;
184                 choix_carte = false;
185                 choix_nb_de = false;
186             }
187
188             //== le joueur a 9 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
189
190             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
191             {
192                 afficher_commentaire_P9, renderer, font, "Vous avez neuf choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
193                 choix_fait = false;
194                 choix_lancer = false;
195                 choix_payer = false;
196                 choix_carte = false;
197                 choix_nb_de = false;
198             }
199
200             //== le joueur a 10 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
201
202             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
203             {
204                 afficher_commentaire_P10, renderer, font, "Vous avez dix choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
205                 choix_fait = false;
206                 choix_lancer = false;
207                 choix_payer = false;
208                 choix_carte = false;
209                 choix_nb_de = false;
210             }
211
212             //== le joueur a 11 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
213
214             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
215             {
216                 afficher_commentaire_P11, renderer, font, "Vous avez onze choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
217                 choix_fait = false;
218                 choix_lancer = false;
219                 choix_payer = false;
220                 choix_carte = false;
221                 choix_nb_de = false;
222             }
223
224             //== le joueur a 12 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
225
226             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
227             {
228                 afficher_commentaire_P12, renderer, font, "Vous avez douze choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
229                 choix_fait = false;
230                 choix_lancer = false;
231                 choix_payer = false;
232                 choix_carte = false;
233                 choix_nb_de = false;
234             }
235
236             //== le joueur a 13 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
237
238             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
239             {
240                 afficher_commentaire_P13, renderer, font, "Vous avez treize choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
241                 choix_fait = false;
242                 choix_lancer = false;
243                 choix_payer = false;
244                 choix_carte = false;
245                 choix_nb_de = false;
246             }
247
248             //== le joueur a 14 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
249
250             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
251             {
252                 afficher_commentaire_P14, renderer, font, "Vous avez quatorze choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
253                 choix_fait = false;
254                 choix_lancer = false;
255                 choix_payer = false;
256                 choix_carte = false;
257                 choix_nb_de = false;
258             }
259
260             //== le joueur a 15 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
261
262             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui_loue-get_p_rison()<0 && joueur_qui_loue-get_nb_de)
263             {
264                 afficher_commentaire_P15, renderer, font, "Vous avez quinze choix pour sortir de prison: Choisissez soit lancer les dés, soit payer 50€ pour
265                 choix_fait = false;
266                 choix_lancer = false;
267                 choix_payer = false;
268                 choix_carte = false;
269                 choix_nb_de = false;
270             }
271
272             //== le joueur a 16 choix : lancer les dés, payer 50€, se faire une carte p_rison, se faire un dé et se faire un dé -> il n'a pas de carte p_rison ==>
273
274             else if (joueur_qui_loue-get_argent()<50 && joueur_qui_loue-get_carte_p_rison()==0 && joueur_qui
```

```

113 //== Le joueur a 2 choix : lancer les dés et payer 50€ pour sortir de prison ==> Il n'a pas de carte prison ==>
114
115 else if (joueur_qui_joue-get_argent())>50 && joueur_qui_joue-get_carte_prison()==0)
116 {
117     afficher_commentaireDemande(P, renderer, font, "Vous avez deux choix pour sortir de prison: Choisissez soit lancer les dés, pour tenter
118
119     choixfait = false;
120     while (choixfait == false)
121     {
122         SDL_PumpEvents(); // Mettre à jour l'état du clavier
123         const Uint8* keystates = SDL_GetKeybaordstate(NULL);
124
125         if (keystates[SDL_SCANCODE_P])
126         {
127             // L'utilisateur a appuyé sur la touche p
128             afficher_commentaire(P, renderer, font, "Vous voulez lancer les dés", 5000);
129             choix_lancer = true;
130             choixfait = true;
131             prison = false;
132             effacer_commentaire(renderer);
133         }
134     }
135
136     else if (keystates[SDL_SCANCODE_P])
137     {
138         // L'utilisateur a appuyé sur la touche p
139         afficher_commentaire(P, renderer, font, " Vous voulez payer 50€", 5000);
140         choix_payer = true;
141         choixfait = true;
142         prison = false;
143         effacer_commentaire(renderer);
144     }
145 }
146
147 //Faire le choix du joueur
148 if (choix_payer)
149 {
150     payer_50(joueur_qui_joue);
151 }
152
153 else if (choix_lancer)
154 {
155     lancer_de_prison(P,joueur_qui_joue, renderer, font);
156 }
157
158 }
159

```

```

153
154
155
156
157
158
159
160 //Le joueur a 2 choix de lancer les dés et utiliser une carte sortir --> Le joueur n'a pas assez d'argent pour payer 50€ mee/
161
162 else if (joueur_qui_joue-get_argent()<50 && joueur_qui_joue-get_carte_prison()>0){
163
164     afficher_commeintaireDes(P, rendre, fait, "Vous avez 2 choix pour sortir de prison ! Choisissez entre utiliser une carte librer de prison appuyer
165
166     choix_fait = false;
167     while (!choix_fait) {
168         SDL_PumpEvents(); // Mettre à jour l'état du clavier.
169         const Uint8* keyStates = SDL_GetKeyboardState(&nLutpr);
170
171         if (keyStates[SDL_SCANCODE_0])
172             // Utilisateur a appuyé sur la touche 0
173             afficher_commeintaire(P, rendre, fait, "Vous voulez lancer les dés", 5000);
174             choix_lancer = true;
175             choix_fait = true;
176             prison = false;
177             effacer_commeintaire(rendre);
178         }
179
180     else if (keyStates[SDL_SCANCODE_C])
181         // Utilisateur a appuyé sur la touche c
182         afficher_commeintaire(P, rendre, fait, "Vous voulez utiliser une carte librer de prison", 5000);
183         choix_carte = true;
184         choix_fait = true;
185         effacer_commeintaire(rendre);
186     }
187
188 }
189
190 //Faire le choix du joueur
191
192 if(choix_carte)
193 {
194     utiliser_carte_prison(joueur_qui_joue);
195 }
196
197 if (choix_lancer)

```

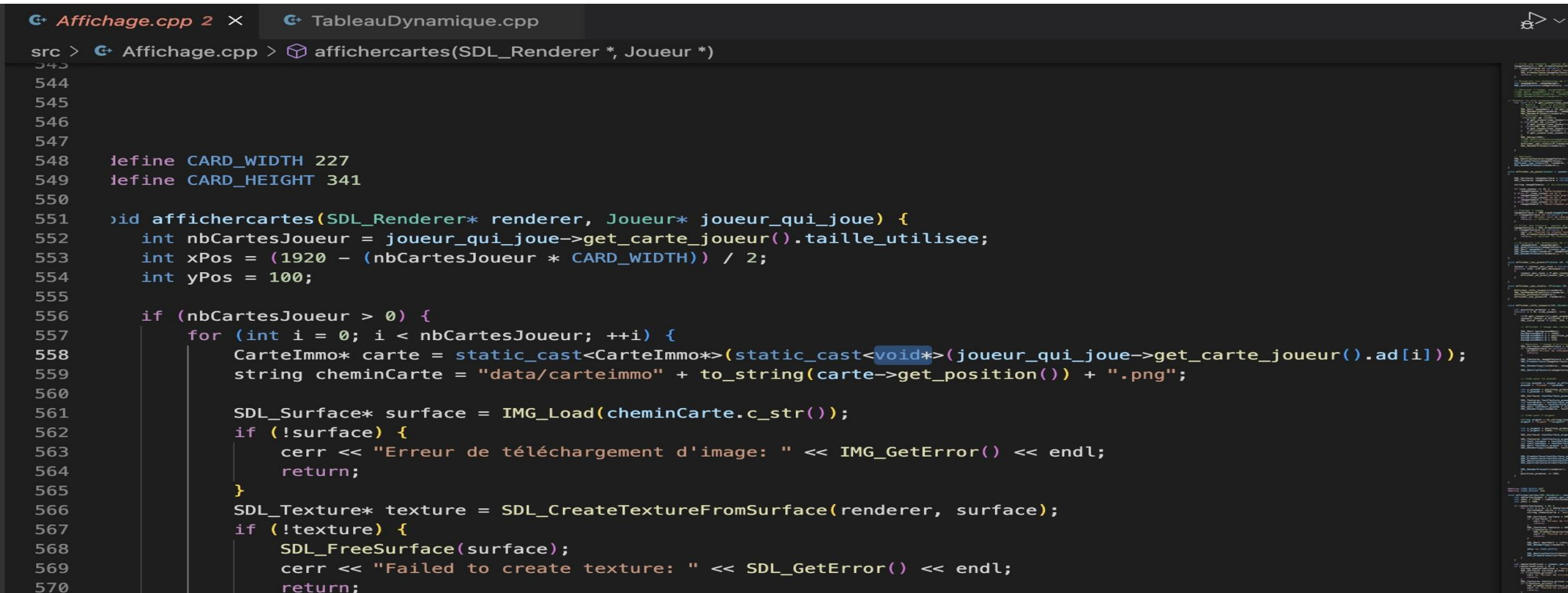
Pointeurs (void *)

- On avait stocker plusieurs données du jeu dans des tableaux dynamiques , comme par exemple les cartes des joueurs (cartes gare, carte immobilières et cartes services publics) . Notre donnée membre ad de la classe TableauDynamique pointe sur ElementTD .
- Comme on avait plusieurs ElementTD de type différents , on pouvait pas associer ElementTD à un seule type . Donc , on a décidé de l'associer à un void* .

```
Joueur.h X TableauDynamique.cpp  
c > C Joueur.h > ...  
4  
5 #include "TableauDynamique.h"  
6 #include <string>  
7 using namespace std;  
8  
9 class Joueur{  
10  
11     private:  
12         string pseudo;  
13         int argent;  
14         int etat_prison;  
15         int position;  
16         TableauDynamique carte_joueur;  
17         TableauDynamique carte_gare; ~/Desktop/Projet 8/src/Chance_fonctions.h  
18         TableauDynamique carte_services_publics;  
19         int carte_prison; //compteur de cartes libéré de prison  
20         int position_x;  
21         int position_y;  
22  
23     public:  
24         Joueur();  
25  
26         string get_pseudo();  
27         int get_argent();  
28  
29         void set_pseudo(string p);  
30         void set_argent(int a);  
31         void set_etat_prison(int e);  
32         void set_position(int pos);  
33         void set_carte_joueur(TableauDynamique c);  
34         void set_carte_gare(TableauDynamique g);  
35         void set_carte_services_publics(TableauDynamique s);  
36  
37         void afficher();  
38         void jouer();  
39         void se_deplacer();  
40         void se_retourner();  
41         void se_ranger();  
42         void se_liberer();  
43         void se_tourner();  
44         void se_dresser();  
45         void se_coucher();  
46         void se_marcher();  
47         void se_sauter();  
48         void se_glisser();  
49         void se_creper();  
50         void se_battre();  
51         void se_voler();  
52         void se_ranger();  
53         void se_liberer();  
54         void se_tourner();  
55         void se_dresser();  
56         void se_coucher();  
57         void se_marcher();  
58         void se_sauter();  
59         void se_glisser();  
60         void se_creper();  
61         void se_battre();  
62         void se_voler();  
63         void se_ranger();  
64         void se_liberer();  
65         void se_tourner();  
66         void se_dresser();  
67         void se_coucher();  
68         void se_marcher();  
69         void se_sauter();  
70         void se_glisser();  
71         void se_creper();  
72         void se_battre();  
73         void se_voler();  
74         void se_ranger();  
75         void se_liberer();  
76         void se_tourner();  
77         void se_dresser();  
78         void se_coucher();  
79         void se_marcher();  
80         void se_sauter();  
81         void se_glisser();  
82         void se_creper();  
83         void se_battre();  
84         void se_voler();  
85         void se_ranger();  
86         void se_liberer();  
87         void se_tourner();  
88         void se_dresser();  
89         void se_coucher();  
90         void se_marcher();  
91         void se_sauter();  
92         void se_glisser();  
93         void se_creper();  
94         void se_battre();  
95         void se_voler();  
96         void se_ranger();  
97         void se_liberer();  
98         void se_tourner();  
99         void se_dresser();  
100        void se_coucher();  
101        void se_marcher();  
102        void se_sauter();  
103        void se_glisser();  
104        void se_creper();  
105        void se_battre();  
106        void se_voler();  
107        void se_ranger();  
108        void se_liberer();  
109        void se_tourner();  
110        void se_dresser();  
111        void se_coucher();  
112        void se_marcher();  
113        void se_sauter();  
114        void se_glisser();  
115        void se_creper();  
116        void se_battre();  
117        void se_voler();  
118        void se_ranger();  
119        void se_liberer();  
120        void se_tourner();  
121        void se_dresser();  
122        void se_coucher();  
123        void se_marcher();  
124        void se_sauter();  
125        void se_glisser();  
126        void se_creper();  
127        void se_battre();  
128        void se_voler();  
129        void se_ranger();  
130        void se_liberer();  
131        void se_tourner();  
132        void se_dresser();  
133        void se_coucher();  
134        void se_marcher();  
135        void se_sauter();  
136        void se_glisser();  
137        void se_creper();  
138        void se_battre();  
139        void se_voler();  
140        void se_ranger();  
141        void se_liberer();  
142        void se_tourner();  
143        void se_dresser();  
144        void se_coucher();  
145        void se_marcher();  
146        void se_sauter();  
147        void se_glisser();  
148        void se_creper();  
149        void se_battre();  
150        void se_voler();  
151        void se_ranger();  
152        void se_liberer();  
153        void se_tourner();  
154        void se_dresser();  
155        void se_coucher();  
156        void se_marcher();  
157        void se_sauter();  
158        void se_glisser();  
159        void se_creper();  
160        void se_battre();  
161        void se_voler();  
162        void se_ranger();  
163        void se_liberer();  
164        void se_tourner();  
165        void se_dresser();  
166        void se_coucher();  
167        void se_marcher();  
168        void se_sauter();  
169        void se_glisser();  
170        void se_creper();  
171        void se_battre();  
172        void se_voler();  
173        void se_ranger();  
174        void se_liberer();  
175        void se_tourner();  
176        void se_dresser();  
177        void se_coucher();  
178        void se_marcher();  
179        void se_sauter();  
180        void se_glisser();  
181        void se_creper();  
182        void se_battre();  
183        void se_voler();  
184        void se_ranger();  
185        void se_liberer();  
186        void se_tourner();  
187        void se_dresser();  
188        void se_coucher();  
189        void se_marcher();  
190        void se_sauter();  
191        void se_glisser();  
192        void se_creper();  
193        void se_battre();  
194        void se_voler();  
195        void se_ranger();  
196        void se_liberer();  
197        void se_tourner();  
198        void se_dresser();  
199        void se_coucher();  
200        void se_marcher();  
201        void se_sauter();  
202        void se_glisser();  
203        void se_creper();  
204        void se_battre();  
205        void se_voler();  
206        void se_ranger();  
207        void se_liberer();  
208        void se_tourner();  
209        void se_dresser();  
210        void se_coucher();  
211        void se_marcher();  
212        void se_sauter();  
213        void se_glisser();  
214        void se_creper();  
215        void se_battre();  
216        void se_voler();  
217        void se_ranger();  
218        void se_liberer();  
219        void se_tourner();  
220        void se_dresser();  
221        void se_coucher();  
222        void se_marcher();  
223        void se_sauter();  
224        void se_glisser();  
225        void se_creper();  
226        void se_battre();  
227        void se_voler();  
228        void se_ranger();  
229        void se_liberer();  
230        void se_tourner();  
231        void se_dresser();  
232        void se_coucher();  
233        void se_marcher();  
234        void se_sauter();  
235        void se_glisser();  
236        void se_creper();  
237        void se_battre();  
238        void se_voler();  
239        void se_ranger();  
240        void se_liberer();  
241        void se_tourner();  
242        void se_dresser();  
243        void se_coucher();  
244        void se_marcher();  
245        void se_sauter();  
246        void se_glisser();  
247        void se_creper();  
248        void se_battre();  
249        void se_voler();  
250        void se_ranger();  
251        void se_liberer();  
252        void se_tourner();  
253        void se_dresser();  
254        void se_coucher();  
255        void se_marcher();  
256        void se_sauter();  
257        void se_glisser();  
258        void se_creper();  
259        void se_battre();  
260        void se_voler();  
261        void se_ranger();  
262        void se_liberer();  
263        void se_tourner();  
264        void se_dresser();  
265        void se_coucher();  
266        void se_marcher();  
267        void se_sauter();  
268        void se_glisser();  
269        void se_creper();  
270        void se_battre();  
271        void se_voler();  
272        void se_ranger();  
273        void se_liberer();  
274        void se_tourner();  
275        void se_dresser();  
276        void se_coucher();  
277        void se_marcher();  
278        void se_sauter();  
279        void se_glisser();  
280        void se_creper();  
281        void se_battre();  
282        void se_voler();  
283        void se_ranger();  
284        void se_liberer();  
285        void se_tourner();  
286        void se_dresser();  
287        void se_coucher();  
288        void se_marcher();  
289        void se_sauter();  
290        void se_glisser();  
291        void se_creper();  
292        void se_battre();  
293        void se_voler();  
294        void se_ranger();  
295        void se_liberer();  
296        void se_tourner();  
297        void se_dresser();  
298        void se_coucher();  
299        void se_marcher();  
300        void se_sauter();  
301        void se_glisser();  
302        void se_creper();  
303        void se_battre();  
304        void se_voler();  
305        void se_ranger();  
306        void se_liberer();  
307        void se_tourner();  
308        void se_dresser();  
309        void se_coucher();  
310        void se_marcher();  
311        void se_sauter();  
312        void se_glisser();  
313        void se_creper();  
314        void se_battre();  
315        void se_voler();  
316        void se_ranger();  
317        void se_liberer();  
318        void se_tourner();  
319        void se_dresser();  
320        void se_coucher();  
321        void se_marcher();  
322        void se_sauter();  
323        void se_glisser();  
324        void se_creper();  
325        void se_battre();  
326        void se_voler();  
327        void se_ranger();  
328        void se_liberer();  
329        void se_tourner();  
330        void se_dresser();  
331        void se_coucher();  
332        void se_marcher();  
333        void se_sauter();  
334        void se_glisser();  
335        void se_creper();  
336        void se_battre();
```


Difficulté : Convertir void* en un type spécifique souhaité

A chaque fois qu'on voulait utiliser les données contenus dans le tableau dynamique, il fallait convertir le type void* pour que ElementTD prenne le type souhaité et pour qu'on puisse récupérer des données ce qui a posé une difficulté vu qu'on ne connaissait pas cette approche.



```
src > Affichage.cpp > affichercartes(SDL_Renderer *, Joueur *)
543
544
545
546
547
548 #define CARD_WIDTH 227
549 #define CARD_HEIGHT 341
550
551 void affichercartes(SDL_Renderer* renderer, Joueur* joueur_qui_joue) {
552     int nbCartesJoueur = joueur_qui_joue->get_carte_joueur().taille_utilisee;
553     int xPos = (1920 - (nbCartesJoueur * CARD_WIDTH)) / 2;
554     int yPos = 100;
555
556     if (nbCartesJoueur > 0) {
557         for (int i = 0; i < nbCartesJoueur; ++i) {
558             CarteImmo* carte = static_cast<CarteImmo*>(static_cast<void*>(joueur_qui_joue->get_carte_joueur().ad[i]));
559             string cheminCarte = "data/carteimmo" + to_string(carte->get_position()) + ".png";
560
561             SDL_Surface* surface = IMG_Load(cheminCarte.c_str());
562             if (!surface) {
563                 cerr << "Erreur de téléchargement d'image: " << IMG_GetError() << endl;
564                 return;
565             }
566             SDL_Texture* texture = SDL_CreateTextureFromSurface(renderer, surface);
567             if (!texture) {
568                 SDL_FreeSurface(surface);
569                 cerr << "Failed to create texture: " << SDL_GetError() << endl;
570                 return;
```

Bibliothèque Json (prochaine diapo : extrait de code)

On a utilisé la bibliothèque JSON pour stocker les informations des cartes immobilières ,

communauté et chance .

- Long à écrire (par exemple dans `carte.json`) .
- Facilite l'organisation hiéarchique des données.
- Facile à manipuler et à modifier (qui aurait pu être plus compliqué si on utilisait un tableau
- dynamique par exemple) .
- Appel fichier Json : déclarer un vecteur de pointeurs qui sera utilisé pour
stocker les données souhaitées . On ouvre le fichier Json , on lit les données
et on les stocke dans l'objet `Json::Value root` '. Pour parcourir les données , on itère sur chaque élément de l'objet root (qui contient les données JSON lues) et on extrait les valeurs nécessaires pour créer les objets du type souhaitée. ON alloue de la

data > {} Carte_chance.json > ...

```
1  [
2    {
3      "id":1,
4      "action": "Avancez jusqu'à la case Départ. Recevez 200€"
5    },
6    {
7      "id":2,
8      "action": "Vous êtes libéré de prison. Cette carte peut être conservée jusqu'à ce qu'elle soit utilisée. "
9    },
10   {
11     "id":3,
12     "action": "Remboursement d'impôt. Recevez 30€."
13   },
14   {
15     "id":4,
16     "action": "Suite à une erreur de l'hôpital, vous perdez vos deux jambes mais vous recevez 100€ et un bon d'achat pour un fauteuil roulant !"
17   },
18   {
19     "id":5,
20     "action": "Payez les frais de scolarité de 50€."
21   },
22   {
23     "id":6,
24     "action": "Recevez votre salaire de 100€."
25   },
26   {
27     "id":7,
28     "action": "Pendant une promenade au parc de la Tête d'or vous trouvez 50€ par terre."
29   },
30   {
31     "id":8,
32     "action": "Vous faites un braquage à main armée sans vous faire attraper (bien joué), recevez 350€."
33   },
34 ]
```

```
24
25 void piocher_carteschances(Plateau &P, Joueur* joueur_qui_joue,SDL_Renderer* renderer, TTF_Font* font){
26
27     vector<Cartechance*> Cartechance;
28     ifstream cartechance("data/Carte_chance.json");
29     if(!cartechance.is_open()){
30         cout << "Impossible d'ouvrir le fichier JSON de cartes chance." <<endl;
31     }
32
33     Json::Value root;
34     Json::CharReaderBuilder reader;
35     Json::parseFromStream(reader, cartechance , &root, nullptr);
36
37     int tab[] = {3, 3, 3, 5, 5, 6, 6, 2, 2, 2, 16, 16, 14, 14, 15, 15, 8, 10, 10, 12, 12, 12, 9, 9, 1, 7, 4, 4, 4, 11, 11, 13};
38     int n = rand()% 31 + 0 ;
39     int ac = tab [n];
40
41     for(const auto& elt : root){
42
43         int i=elt["id"].asInt();
44
45         if (i==ac && i==1){ aller_case_départ_2(joueur_qui_joue);  afficher_carte_chance(P,renderer,font, i);}
46
47         else if (i==ac && i==2){ libéré_de_prison_2(joueur_qui_joue);  afficher_carte_chance(P,renderer,font, i);}
48
49         else if (i==ac && i==3){ remboursement_impot(joueur_qui_joue); afficher_carte_chance(P,renderer,font, i);}
50
51         else if (i==ac && i==4){ erreur_hopital(joueur_qui_joue);  afficher_carte_chance(P,renderer,font, i);}
52     }
```

```

src > Demander_pseudo.cpp > ...
1  #include <gtkmm.h>
2  #include <iostream>
3  #include <vector>
4
5  #include "Joueur.h"
6  #include "Plateau.h"
7
8  class DialogPseudosJoueurs : public Gtk::Dialog {
9  public:
10     DialogPseudosJoueurs(int nbJoueurs) {
11         set_title("Entrer les pseudos des joueurs");
12         set_default_size(300, 150);
13
14         Gtk::Box* boite = Gtk::manage(new Gtk::Box(Gtk::ORIENTATION_VERTICAL, 10));
15         get_content_area()->pack_start(*boite, Gtk::PACK_SHRINK, 0);
16
17         for (int i = 0; i < nbJoueurs; ++i) {
18             Gtk::Entry* champSaisie = Gtk::manage(new Gtk::Entry());
19             champSaisie->set_placeholder_text("Pseudo du joueur " + std::to_string(i + 1));
20             champsSaisie.push_back(champSaisie);
21             boite->pack_start(*champSaisie, Gtk::PACK_SHRINK, 0);
22         }
23
24         add_button("Valider", Gtk::RESPONSE_OK);
25         show_all_children();
26     }
27
28     std::vector<std::string> obtenirPseudosJoueurs() const {
29         std::vector<std::string> pseudos;
30         for (const auto& champSaisie : champsSaisie) {
31             pseudos.push_back(champSaisie->get_text());
32         }
33         return pseudos;
34     }
35
36 private:
37     std::vector<Gtk::Entry*> champsSaisie;
38 };

```

```

src > Demander_pseudo.cpp > ...
36 private:
37     std::vector<Gtk::Entry*> champsSaisie;
38 };
39
40 std::vector<std::string> demanderPseudosJoueurs(int nbJoueurs) {
41     DialogPseudosJoueurs dialogue(nbJoueurs);
42     dialogue.run();
43     return dialogue.obtenirPseudosJoueurs();
44 }
45
46 void entrer_les_pseudos(int nb_joueurs, Plateau &P)
47 {
48     vector<Joueur*> joueurs;
49
50     std::vector<std::string> pseudos = demanderPseudosJoueurs(nb_joueurs);
51
52     for (const auto& pseudo : pseudos) {
53         Joueur *jou = new Joueur(pseudo);
54         jou->set_pseudo(pseudo);
55         joueurs.push_back(jou);
56     }
57
58     for (Joueur *joueur : joueurs) {
59         P.ajouter_joueur(joueur);
60     }
61 }
62
63
64
65 /*int main(int argc, char* argv[])
66 {
67     auto app = Gtk::Application::create(argc, argv, "org.gtkmm.example");
68     Plateau P(0);
69     entrer_les_pseudos(2,P);
70     cout<<P.get_joueur(0)->get_pseudo()<<endl;
71 }*/

```

Nouvelle bibliothèque : gtkM pour récupérer les pseudos des joueurs qui n'était pas possible avec sdl .

Sdl : (vidéo , problème de fluidité dans l'affichage , fonctions complexes...)

- Bien qu'on a codé avec SDL auparavant , lors de la réalisation de notre projet, on a fait face à beaucoup d'aspects inconnus dans le langage . On prend par exemple un de ces aspects ; pour un rendu plus beau et plus fluide , nous avons décidé d'inclure une vidéo de dé qui s'affiche à chaque lancée de dé . Il fallait que nous rencontrions plusieurs erreurs de segmentation pour qu'on réalise que sdl ne prend pas de fichier mp4 et qu'il prend un fichier **BMP** ; le fichier contient une succession d'images qui forment la vidéo et qui est chargé séquentiellement .
(Code dans la diapo suivante)

```

6 // Charger et afficher séquentiellement les images
7 for (int i = 1; i <= 59; ++i) {
8     // Charger l'image courante
9     string cheminImage = "data/dice/000" +to_string(i) + ".bmp";
10    SDL_Surface *surface = SDL_LoadBMP(cheminImage.c_str());
11    if (surface == nullptr) {
12        cerr << "Impossible de charger l'image " << cheminImage << " ! Erreur SDL : " << SDL_GetError() << endl;
13        continue;
14    }
15
16    SDL_Texture *texture = SDL_CreateTextureFromSurface(rendu, surface);
17    SDL_FreeSurface(surface);
18    if (texture == nullptr) {
19        cerr << "Impossible de créer la texture pour l'image " << cheminImage << " ! Erreur SDL : " << SDL_GetError() <<endl;
20        continue;
21    }
22
23    // Effacer l'écran
24    SDL_RenderClear(rendu);
25
26    // Afficher l'image courante
27    SDL_RenderCopy(rendu, texture, NULL, NULL);
28
29    // Mettre à jour l'écran
30    SDL_RenderPresent(rendu);
31
32    // Attendre un peu avant de passer à l'image suivante (100 millisecondes)
33    SDL_Delay(100);
34
35

```


2ème aspect qui a posé problème : Redessin de toute la fenêtre à chaque changement

Vu que c'est impossible d'enlever juste un element de la fenêtre SDL et laisser le reste intact , la fenêtre se redessinait après chaque action graphique ce qui rendait l'affichage laid et lent.

Exemple , Dessin du rectangle ou s'affichait les commentaires.

Au lieu de supprimer toute la fenêtre et la redessiner ,

```
150 // Effacer le rendu en redessinant la fenêtre sans le texte du commentaire
151 //Nettoyer
152 SDL_RenderClear(renderer);
153 afficher_jeu_static(P,renderer, font);
154 SDL_RenderPresent(renderer);
155 }
156
```

On a opté pour coder une fonction (effacer_commentaire)qui dessine un rectangle blanc vide et l'appeler à la fin du code de notre fonction qui affiche le commentaire au lieu d'un RenderClear.

```
void effacer_commentaire(SDL_Renderer* renderer) {  
    // Dessiner un rectangle blanc pour effacer le commentaire précédent  
  
    SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255); // Couleur blanche  
    SDL_Rect rect = {1040, 900, 900, 200};  
    SDL_RenderFillRect(renderer, &rect);
```

```
    // Effacer le rendu en redessinant la fenêtre sans le texte du commentaire  
    //Nettoyer  
    effacer_commentaire(renderer);  
    afficher_jeu_static(P, renderer, font);  
    SDL_RenderPresent(renderer);
```

```
}
```

Fonctions d’affichage : des cartes immobilières possédées et de cartes chances et cartes communautaires quand le joueur pioche une

- Ces fonctions étaient un peu compliquées (par exemple pour la fonction afficher carte chance quand on pioche) ; On a codé une procédure piocher_carte_chances ; on crée un tableau pour répartir les id des cartes chances (il fallait pas que la quantité soit égale ; par exemple on mettrait moins de cartes chances ou il y avait une récompense de 350€)et ensuite on parcourt le fichier json “cartechance.json” et on correspond chaque id (du fichier json) à chaque valeur du tableau et on appelle la fonction qui correspond à la bonne action (définie dans chance_fonction.h) , ainsi que la fonction qui affiche la carte piochée à l’écran (le lien est fait par le id ; toutes les images sont nommées cartechance1.png , cartechance2.png , etc) .

```

#ifndef AUTRES_CARTES_FONCTIONS_H
#define AUTRES_CARTES_FONCTIONS_H

#include "Joueur.h"

// Déclaration des fonctions des cartes
void aller_case_départ_2(Joueur* joueur_who_joue);
void libéré_de_prison_2(Joueur* joueur_who_joue);
void remboursement_impot(Joueur* joueur_who_joue);
void erreur_hopital(Joueur* joueur_who_joue);
void frais_scolarité(Joueur* joueur_who_joue);
void salaire(Joueur* joueur_who_joue);
void par_terre(Joueur* joueur_who_joue);
void braquage(Joueur* joueur_who_joue);
void nautilus(Joueur* joueur_who_joue);
void BU(Joueur* joueur_who_joue);
void dentiste(Joueur* joueur_who_joue);
void code_route(Joueur* joueur_who_joue);
void soudoyez(Joueur* joueur_who_joue);
void anniversaire(Joueur* joueur_who_joue);
void vol_passeport(Joueur* joueur_who_joue);
void voyage(Joueur* joueur_who_joue);

#endif // AUTRES_CARTES_FONCTIONS_H

```

```

//AFFICHER CARTE CHANCE
void afficher_carte_chance(Plateau &P, SDL_Renderer* renderer, TTF_Font* font, int id_carte) {
    //construire le chemin vers l'image de la carte
    string imageChemin = "data/chance" + to_string(id_carte) + ".png";

    // Charger l'image
    SDL_Surface* surface = IMG_Load(imageChemin.c_str());
    if (!surface) {
        cerr << "Erreur de téléchargement d'image: " << IMG_GetError() << endl;
        return;
    }

    // Créer une texture à partir de la surface
    SDL_Texture* texture = SDL_CreateTextureFromSurface(renderer, surface);
    if (!texture) {
        SDL_FreeSurface(surface);
        cerr << "Failed to create texture: " << SDL_GetError() << endl;
        return;
    }

    // Calculer la position et la taille pour centrer l'image
    int x = (1030 - surface->w) / 2;
    int y = (1030 - surface->h) / 2;
    SDL_Rect destRect = {x, y, surface->w, surface->h};

    // Libérer la surface dès qu'elle n'est plus nécessaire
    //SDL_FreeSurface(surface);

    // Afficher l'image
    //SDL_RenderClear(renderer);
    SDL_RenderCopy(renderer, texture, NULL, &destRect); // Copier la texture au rendu
    SDL_RenderPresent(renderer); // Mettre à jour l'écran avec le rendu

    // Attendre 5 secondes
}

```

```

void piocher_carteschances(Plateau &P, Joueur* joueur_who_joue, SDL_Renderer* renderer, TTF_Font* font){

    vector<Cartechance> Cartechance;
    ifstream cartechance("data/Carte_chance.json");
    if(!cartechance.is_open()){
        cout << "Impossible d'ouvrir le fichier JSON de cartes chance." <<endl;
    }
    Json::Value root;
    Json::CharReaderBuilder reader;
    Json::parseFromStream(reader, cartechance , &root, nullptr);

    int tab[] = {3, 3, 3, 5, 5, 6, 6, 2, 2, 2, 16, 16, 14, 14, 15, 15, 8, 10, 10, 12, 12, 12, 9, 9, 1, 7, 4, 4, 4, 11, 11, 13};
    int n = rand()% 31 + 0 ;
    int ac = tab [n];

    for(const auto& elt : root){

        int i=elt["id"].asInt();

        if (i==ac && i==1){ aller_case_départ_2(joueur_who_joue); afficher_carte_chance(P,renderer,font, i);}

        else if (i==ac && i==2){ libéré_de_prison_2(joueur_who_joue); afficher_carte_chance(P,renderer,font, i);}

        else if (i==ac && i==3){ remboursement_impot(joueur_who_joue); afficher_carte_chance(P,renderer,font, i);}

        else if (i==ac && i==4){ erreur_hopital(joueur_who_joue); afficher_carte_chance(P,renderer,font, i);}

        else if (i==ac && i==5){ frais_scolarité(joueur_who_joue); afficher_carte_chance(P,renderer,font, i);}

        else if (i==ac && i==6){ salaire(joueur_who_joue); afficher_carte_chance(P,renderer,font, i);}
    }
}

```

```

cdu_etec = false;

afficher_commentaire_demande(P, renderer, font, "Voulez-vous afficher vos cartes immobilières ? O ou N ?");

bool choix_fait = false;
bool oui = false, non = false;
while (!choix_fait) {
    SDL_PumpEvents(); // Mettre à jour l'état du clavier
    const Uint8* keystates = SDL_GetKeyboardState(nullptr);
    if (keystates[SDL_SCANCODE_0])
    {
        // L'utilisateur a appuyé sur la touche 0
        choix_fait = true;
        oui = true;
        effacer_commentaire(renderer);
    }
    else if (keystates[SDL_SCANCODE_N])
    {
        // L'utilisateur a appuyé sur la touche N
        choix_fait = true;
        non = true;
        effacer_commentaire(renderer);
    }
}

// Effacer le commentaire quand le joueur a fait son choix
afficher_jeu_static(P, renderer, font);
SDL_RenderPresent(renderer);

/** Il veut acheter la carte **/
if (oui)
{
    SDL_RenderClear(renderer);
    SDL_RenderPresent(renderer);
    afficher_cartes(renderer, joueur_qui_joue);
    SDL_RenderPresent(renderer);
    SDL_Delay(10000);
    SDL_RenderClear(renderer);
    SDL_RenderPresent(renderer);
    afficher_jeu_static(P, renderer, font);
}

```

```

#define CARD_WIDTH 227
#define CARD_HEIGHT 341

void afficher_cartes(SDL_Renderer* renderer, Joueur* joueur_qui_joue) {
    int nbCartesJoueur = joueur_qui_joue->get_carte_joueur().taille_utilisee;
    int xPos = (1920 - (nbCartesJoueur * CARD_WIDTH)) / 2;
    int yPos = 100;

    if (nbCartesJoueur > 0) {
        for (int i = 0; i < nbCartesJoueur; ++i) {
            CarteImmo* carte = static_cast<CarteImmo*>(static_cast<void*>(joueur_qui_joue->get_carte_joueur().ad[i]));
            string cheminCarte = "data/carteimmo" + to_string(carte->get_position()) + ".png";

            SDL_Surface* surface = IMG_Load(cheminCarte.c_str());
            if (!surface) {
                cerr << "Erreur de téléchargement d'image: " << IMG_GetError() << endl;
                return;
            }
            SDL_Texture* texture = SDL_CreateTextureFromSurface(renderer, surface);
            if (!texture) {
                SDL_FreeSurface(surface);
                cerr << "Failed to create texture: " << SDL_GetError() << endl;
                return;
            }

            SDL_Rect destRect = {xPos, yPos, CARD_WIDTH, CARD_HEIGHT};
            SDL_RenderCopy(renderer, texture, NULL, &destRect);

            xPos += CARD_WIDTH;

            SDL_DestroyTexture(texture);
            SDL_FreeSurface(surface);
        }
    }
}

```

Afficher les cartes immobilières possédées

Graphique , Design : GIMP

Éditer le plateau Monopoly version lyon 1, les cartes chances , les cartes communaitaires et les cartes immobilières ; pas compliqué mais très long.



Caisse de communauté



TITRE DE PROPRIÉTÉ	
Omega 11	
Loyer	¥26
Avec groupe complet	¥52
Avec 1	¥130
Avec 2	¥390
Avec 3	¥900
Avec 4	¥1100
Avec 5	¥1275
Prix des maisons ¥200 chacune	
Prix d'1 hôtel ¥200 chacun (plus 4 maisons)	
© 1935, 2021 HASBRO.	

La Doua Gaston Berger	
LOYER	
Si vous possédez 2 Gares	¥50
Si vous possédez 3 Gares	¥100
Si vous possédez 4 Gares	¥200
© 1935, 2021 HASBRO.	

Conclusion

- Objectifs initiaux atteints ? : Nos objectifs sont plus ou moins atteints , On avait loupé certaines règles secondaires du jeu qu'on avait pas le temps de faire. On voulait aussi faire plus d'animations et optimiser plus la mémoire vu qu'il ya beaucoup de fonctions trop longues et répétitives ou bien même pour le remplacement du `RenderClear` par un autre dessin pour éviter les flashes(c'était la seule solution qu'on avait trouvé).
- Difficultés persistentes : Quitter la fenêtre `SDL` pendant une boucle d'actions , on était obligées à envoyer un signal `KILL` au processus à chaque fois qu'on voulait quitter le jeu , on avait eu l'idée de coder un gestionnaire de signal qui envoie le signal `KILL` au processus à chaque fois on clique sur la croix (`SDL_Quit`) ; on pouvait quitter que lorsqu'il y'avait pas une action courante . On aurait pu s'approfondir dans l'idée si on avait plus du temps.