Sortera includes bättre:
Ordningen brukar vara
#include <standardklass i c>
#include <standardklass i c++>

#include <tredjeparts-header>

#include "header från samma mapp"

Så detta:
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include <map>
#include "block.h"

Bör hellre vara
#include <map>

#include <SDL/SDL.h>
#include <SDL/SDL_image.h>

#include "block.h"

Var mer konsistent med namngivning av privata variabler:
I PlayState.h så heter dom rock, world, indestruct
I alla andra headerfiler så slutar dom med understreck, e.g. keystate_

Namnge gärna metoder tydligare, eller mer som verb
Exempel:
void Player::angle_GL(bool);
  Vafalls!?
bool Entity::collision_entity(Entity*)
 Det är lite vagt vad metoden egentligen gör. Kollar den om två Entities kolliderar?
 Modifierar den två Entities som precis har kolliderat? Vad skulle True/False som returvärde innebära?
 Bättre exempel skulle kunna vara bool Entity::hasCollidedWith(Entity*) eller Entity::handleCollisionWith(Entity*)
Ser nog snyggare och mer logiskt ut om ni överlagrar collision istället för att ha en collision_entity, collision_block, collision_blah

Sätt funktioner som inte förändrar klassen till const
Exempel en get_texture()-metod som gör "return texture_;" får gärna vara const

Akta er för att returnera mutable element från en klass.
Entity har följande privata variabel
SDL_Rect sheet_;
och följande publika metod
SDL_Rect& get_sheet();
Ifall man nu skriver Entity.get_sheet().w = 4; så kommer det privata värdet att ändras!
Bättre: const SDL_Rect &get_sheet();
Bäst: const SDL_Rect &get_sheet() const;

Akta er för att skriva över pekare utan att frige den först!
I GameEngine.game():
Rad 22: SDL_Surface* screen = nullptr;
Rad 29: screen = SDL_SetVideoMode(...);
Rad 81: screen = SDL_SetVideoMode(...);
Rad 82: SDL_Quit();
screen från rad 29 friges aldrig (om jag inte missat något!)

Använd inte C-casts, utan istället c++-casts
PlayState.cpp: rad 75:
player -> set_speed((short int)3, (short int)0);
I C++ skriver man:
player-> set_speed(dynamic_cast<short int>(3), dynamic_cast<short int>(0));

Skriv gärna mutable variabler med små bokstäver och const med caps:
PlayState.cpp: rad 111:
SDL_Rect BLOCK_ITEM
bör nog heta SDL_Rect block_item eller const SDL_Rect BLOCK_ITEM

Försök radbryta väldigt breda rader
Standard är ofta max 80 tecken per rad. Vissa företag/organisationer kör längre rader, men 236 tecken på en rad är definitivt för mycket! (Player.cpp: rad 12)
Förutom i Player.cpp har ni fixat det gallant!

Lite konstigt arv av Sprite-klasser
Ni har Entity vilket jag antar är eran Sprite-klass, men Block (som är ett grafiskt block) är inte en del av den klassen, utan endast Player är. Tvärtom så inkluderar Entity Block för att fungera, vilket är lite konstigt.

Bra:
Ni har satt headerguards överallt
Bra konsistent indentation (Undantag PlayState.cpp rad 50-54)
Bra konsistent namngivna metoder
Bra att ni har gjort alla variabler privata, med metoder som kallar på dom (undantag Block.size, som är const). Jag tror detta är okej, men helst inte!
Bra kommentarer i källkodsfilerna! Tyvärr inga kommentarer i headers, men det är förståeligt ;)
Bra, tydlig namngivning av de flesta privata variabler! Kunde förstå samtliga i Player.h efter första anblicken!
Mycket bra struktur på koden! Kod som hör samman är grupperade i block, och det är lagom mycket "luft" mellan raderna så man ser bra.
Intressant lösning med olika states som ni växlar mellan (har inte sett det förut)